TensorFlow Performance Optimizations on Intel Architectures

ACLF Developer Session
25 July, 2018

Vamsi Sripathi, AI & HPC Performance Engineer

Vikram Saletore, Ph.D. Principal Engineer

Customer Solution Enablement, AI Products Group, Intel Corp

Representing the work of several Intel teams

Agenda

- Motivation
- TensorFlow Optimizations
  - MKL-DNN
  - Graph optimizations
- Distributed TensorFlow
- Performance Results
- Using Intel TensorFlow
  - Installation
  - Run-time Settings
  - Profiling
  - Potential Issues
- Benchmarks & Case Studies

Motivation

- TensorFlow
  - Popular open-source machine learning/deep learning framework
  - Front end wrapper in Python, core backend in C++
  - Multi node support
  - Widely used in industry for text, speech, image classification
  - Gaining popularity among scientific community (high energy physics, climate)
- Intel Architectures
  - Xeon: Skylake (AVX512)
    - Up to 56 cores/112 threads, DDR memory
  - Xeon Phi: Knight Landing (AVX512)
    - 68 cores/272 threads
    - High bandwidth MCDRAM (16 GB)
- To maximize performance, optimize TensorFlow for Intel hardware
  - Vectorization (FMA unit utilization)
  - Loop blocking (Cache locality/reuse)
  - Parallelization (load balancing)

		AVX2	AVX512
Vector Register Length		256 bits	512 bits
# of FMA's per cycle		2	2
Single Precision	# of FP elements per register	8	16
	Flops per cycle	32	64
Double Precision	# of FP elements per register	4	8
	Flops per cycle	16	32

TensorFlow: Baseline Vs Intel

TensorFlow Baseline	Intel Optimized TensorFlow	
Python: ML/DL Application	Python: ML/DL Application	Frontend
C++: Data Flow Graph	C++: Intel tuned data flow graph	Backend
Eigen Library (Convolutions, Activation Functions, GEMM etc)	Intel MKL-DNN (Convolutions, Activation Functions, GEMM etc)	
Hardware (CPU, GPU)	Intel CPU's (Broadwell, Skylake, Knights Landing)	

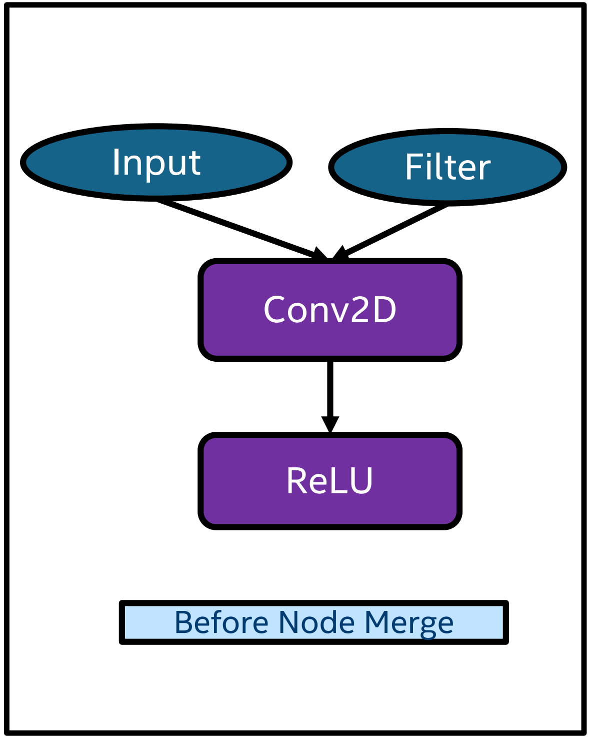
MKL-DNN Optimizations

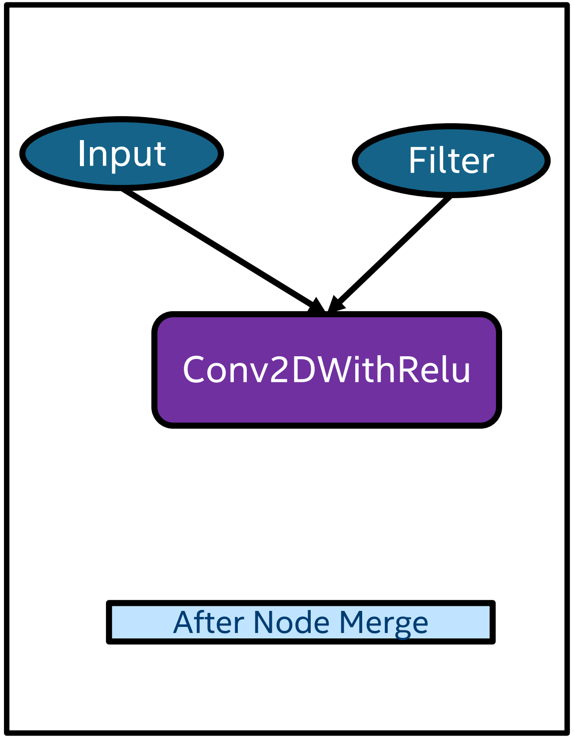
- Eigen implementation of DNN kernels is sub-optimal for Intel hardware
- Intel MKL-DNN is highly optimized, open-source ML/DL library for Intel CPU's
  - Specialized assembly-like kernels for DNN primitives
  - Dedicated kernels based on ISA/hardware architecture (SSE4.2, AVX2, AVX512)
  - OpenMP based multi-threading
  - https://github.com/intel/mkl-dnn
- Replaced Eigen API's in TensorFlow C++ backend with MKL-DNN

Primitives	**Class**
Convolution Deconvolution Inner Product Vanilla RNN, LSTM, GRU	Compute intensive operations
Pooling AVG/MAX Batch Normalization LRN Activations (ReLU, Tanh, ELU, Softmax, …) Sum	Memory bandwidth limited operations
Reorder Concatenation	Data manipulation

Graph Optimizations: Operator Fusion

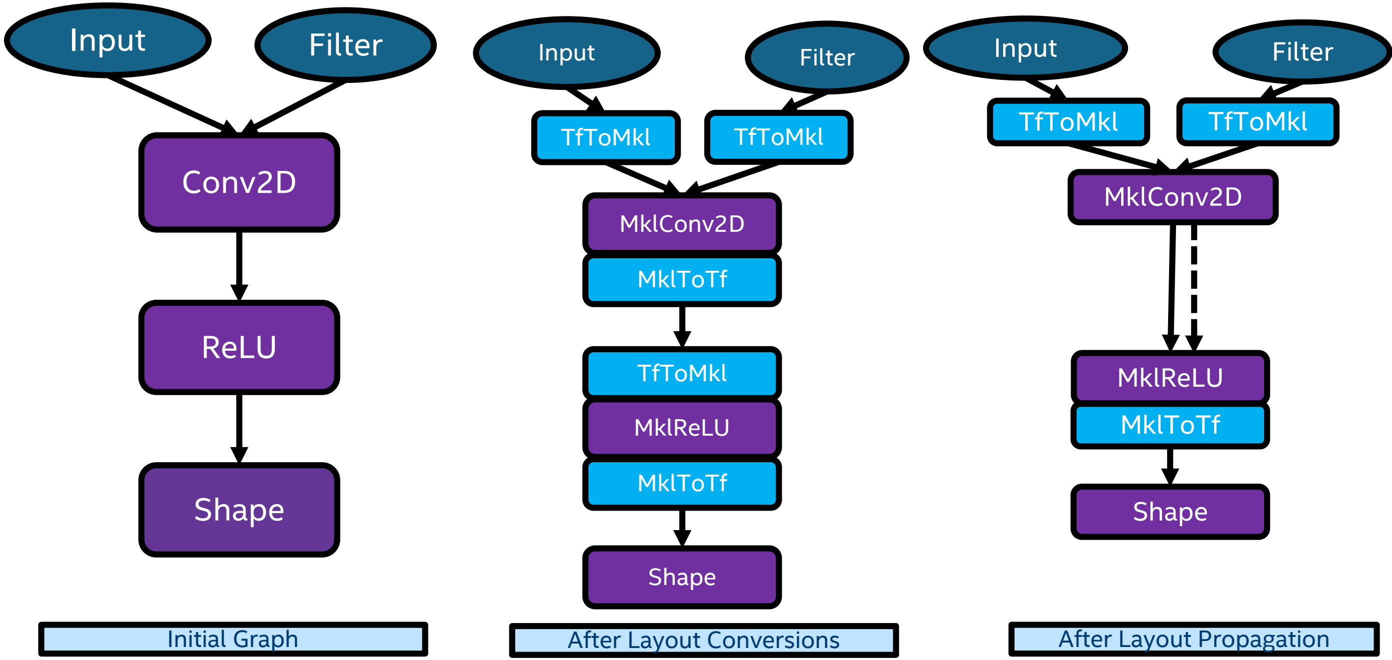
- Popular DNN topologies spent significant amount of time in bandwidth-bound ops
- Fuse BW-bound operators with compute ops to reduce memory pressure





Graph Optimizations: Memory Layout

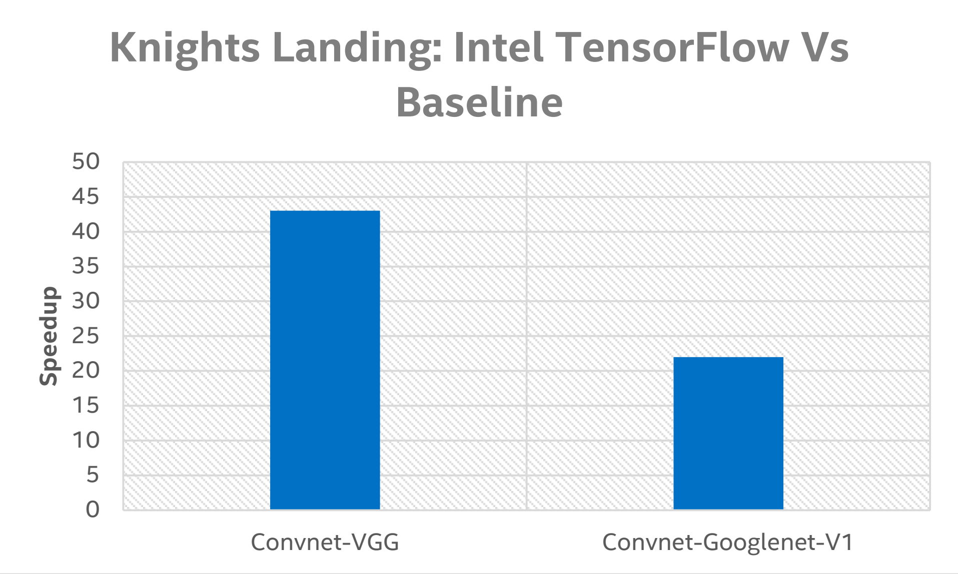
- Most popular memory layouts for image recognition are **nhwc** and **nchw**
- MKL-DNN convolutions use blocked layouts (e.g. nChw16c for AVX512)
- TensorFlow tracks memory layouts and perform reorders **only** when necessary

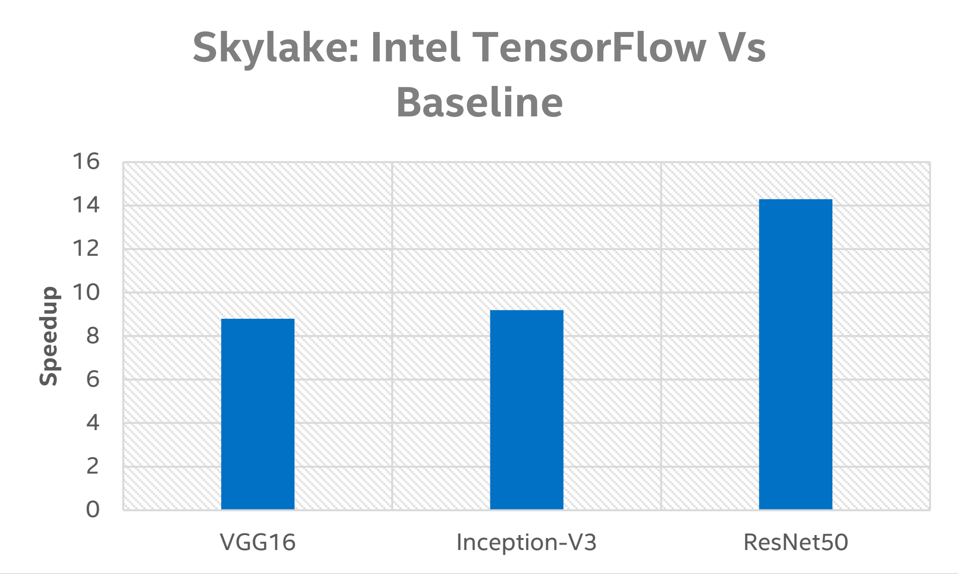


Memory Manager

- Neural network operators (Convolutions, Matrix Multiplications) allocate large chunks of memory
- TensorFlow default memory management routines did not handle the frequent alloc/dealloc
- Implemented a memory pool allocator that reduces the overhead of memory management

Single-Node Performance

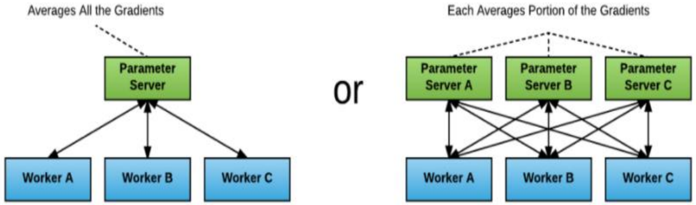




Distributed TensorFlow

- Data Parallelism: Run the same model on all nodes with different data
- DL training is a strong-scaling problem
- Two mechanisms
  - Master-Worker (Google Remote Procedure Call [gRPC])
  - MPI (Uber Horovod, Intel Machine Learning Scaling Library [MLSL])
- gRPC: Inefficient scaling due to bottlenecks at parameter servers
- MPI
  - Horovod: Overlap communication and computation
  - MLSL: Horovod with better MPI_Allreduce() – WIP, Expect

With Parameter Server



Uber's open source MPI based Distributed training framework for TensorFlow

https://github.com/intel/MLSL

Intel TensorFlow: Installation

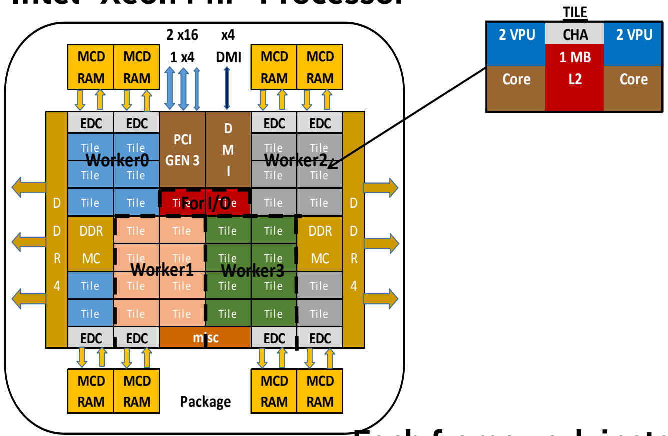
- All Intel optimizations to TensorFlow are upstreamed regularly -- https://github.com/tensorflow
- Get pre-compiled binaries
  - Using pip:
    - Python 2.7: `pip install https://anaconda.org/intel/tensorflow/1.6.0/download/tensorflow-1.6.0-cp27-cp27mu-linux_x86_64.whl`
    - Python 3.5: `pip install https://anaconda.org/intel/tensorflow/1.6.0/download/tensorflow-1.6.0-cp35-cp35m-linux_x86_64.whl`
    - Python 3.6: `pip install https://anaconda.org/intel/tensorflow/1.6.0/download/tensorflow-1.6.0-cp36-cp36m-linux_x86_64.whl`
- Build from source
  - `$ git clone https://github.com/tensorflow/tensorflow.git`
  - `$ cd tensorflow`
  - `$ ./configure`
  - `$ bazel build --config=opt --config=mkl //tensorflow/tools/pip_package:build_pip_package`
  - `$ bazel-bin/tensorflow/tools/pip_package/build_pip_package ~/path_to_wheel_dir`
  - `$ pip install --upgrade --user ~/path_to_wheel_dir/<wheel_name.whl>`

Intel TensorFlow: Run-time Settings

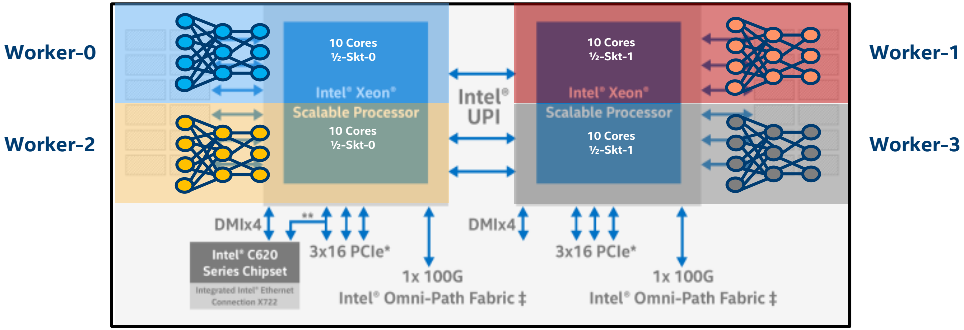
- Threading
  - inter_op_parallelism_threads = max number of operators that can be executed in parallel.
  - intra_op_parallelism_threads = max number of threads executing an operator. Usually set to # of physical cores or logical cores assigned for TensorFlow instance
  - OMP_NUM_THREADS = max number of threads executing MKL functions. Usually set to # of physical cores assigned for TensorFlow instance
  - KMP_AFFINITY = granularity=fine,compact,1,0
  - KMP_BLOCKTIME =1 or 30
  - https://www.tensorflow.org/performance/performance_guide#optimizing_for_cpu
- Memory
  - If MCDRAM is in Flat-mode and app memory is < 16GB, use numactl –m0 python train.py
- Multiple TensorFlow workers on a single KNL

Training: Multiple TensorFlow Workers per Socket

Intel® Xeon Phi™ Processor



2S Intel® Xeon® Processor



Each framework instance is affinitized to cores or NUMA domains

Each CPU running 1 or more workers/node

Uses optimized MPI library for gradient updates over shared memory

Caffe – Use Optimized Intel® MPI ML Scaling Library (ML-SL)

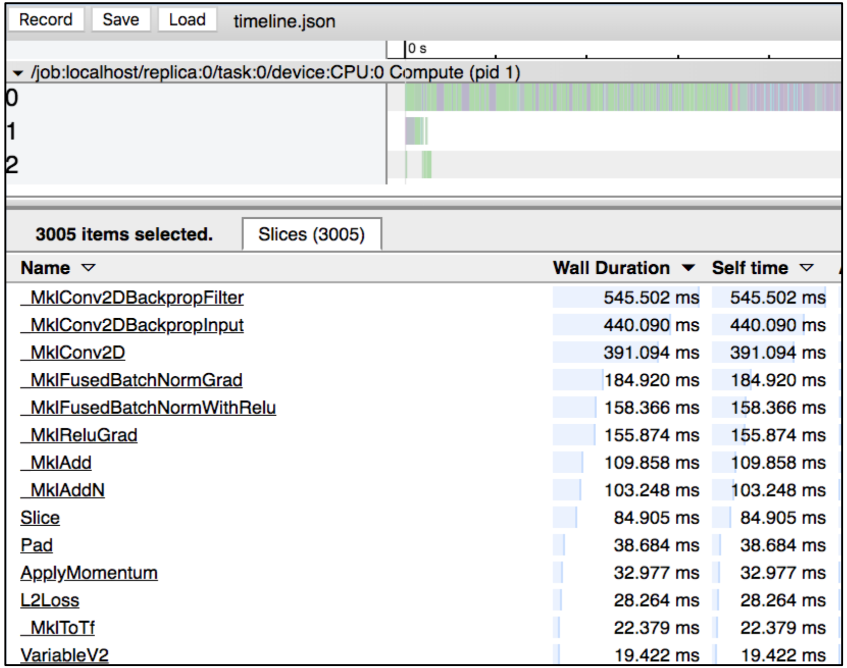
TensorFlow – Uber horovod MPI Library

Optimizations at run time <u>without</u> framework code change

- Intel Best Known Methods
  - https://software.intel.com/en-us/articles/boosting-deep-learning-training-inference-performance-on-xeon-and-xeon-phi
  - https://ai.intel.com/accelerating-deep-learning-training-inference-system-level-optimizations/

Intel TensorFlow: Profiling

- TensorFlow Timeline API's [1]
- Intel MKL-DNN Verbose mode
  - export MKLDNN_VERBOSE=1
  - export MKL_VERBOSE=1
- Intel VTune Analyzer



[1]: https://towardsdatascience.com/howto-profile-tensorflow-1a49fb18073d

Intel TensorFlow: Potential Issues

- Threading
  - Incorrect setting of threading model parameters can lead to over- or under-subscription, leading to poor performance
- Large number of MKL Reorder ops
- Non-multiples of 16 output channels in convolutions
- Non-MKL operators
- I/O pipeline

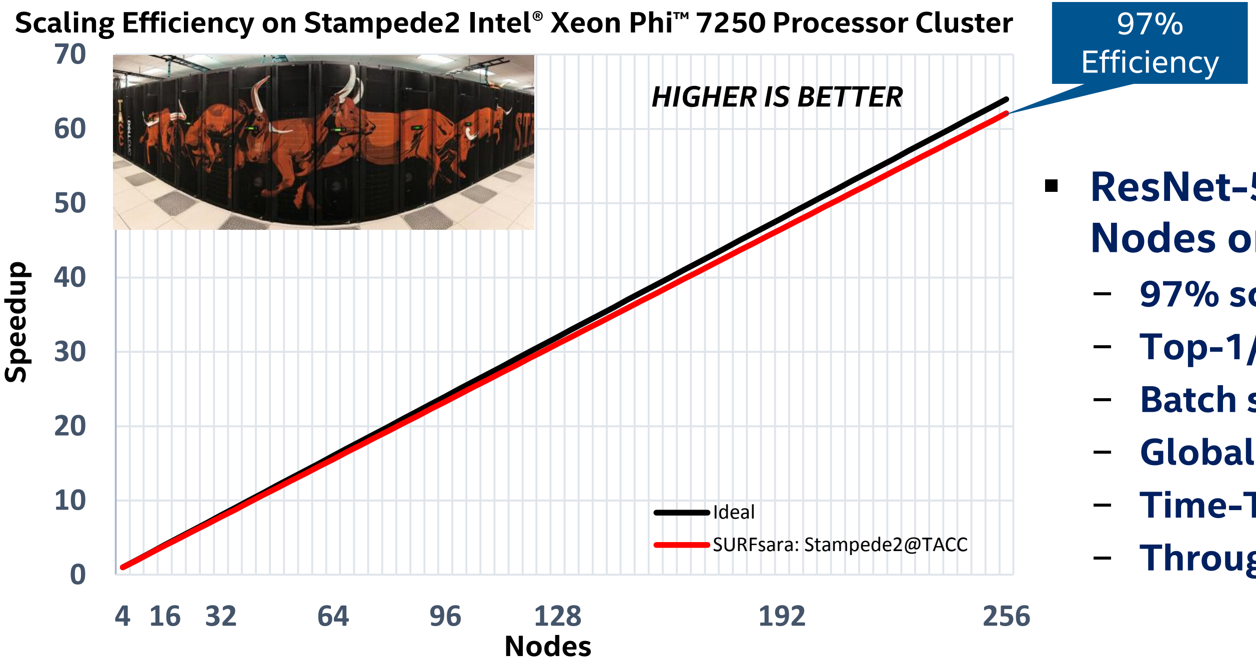
```
OMP: Error #34: System unable to
allocate necessary resources for OMP
thread:

OMP: System error #11: Resource
temporarily unavailable

OMP: Hint: Try decreasing the value of
OMP_NUM_THREADS.
```

SCALEOUT TRAINING PERFORMANCE WITH BENCHMARKS & USE CASES

Multi-Node ResNet-50 Scaling Efficiency

*on **Intel® Xeon Phi™ 7250 Processor** Cluster Stampede2* at TACC**

Scaling Efficiency on Stampede2 Intel® Xeon Phi™ 7250 Processor Cluster



- **ResNet-50 with ImageNet-1K on 256 Nodes on Stampede2/TACC***
  - **97% scaling efficiency**
  - **Top-1/Top-5 > 74%/92%**
  - **Batch size of 16 per node**
  - **Global BS=4096**
  - **Time-To-Train: 63 minutes (37 Epochs)**
  - **Throughput: 12526 Images/sec**

***TACC** (Texas Advanced Computing Center): https://www.tacc.utexas.edu/*

ResNet-50 Training Time to 74% Top-1 Accuracy

*on Intel® Xeon Phi™ 7250 Processor Cluster Stampede2 at TACC**

Intel® Distribution of Caffe* with ImageNet-1K dataset

TRAINING TIME	TRAINING ACCURACY
46 Minutes	74.0% Top-1
LOWER IS BETTER	*HIGHER IS BETTER*

512 Intel® Xeon Phi™ Processor Nodes

Global BS= 12288 & 54 Epochs
24932 Images/sec

TRAINING TIME	TRAINING ACCURACY
39 Minutes	74.2% Top-1
LOWER IS BETTER	*HIGHER IS BETTER*

768 Intel® Xeon Phi™ Processor Nodes

Global BS= 12288 & 61.5 Epochs
33608 Images/sec

***TACC** *(Texas Advanced Computing Center): https://www.tacc.utexas.edu/*

ResNet-50 Training Time to 75.5% Top-1 Accuracy

*on Intel® Xeon Phi™ 7250 Processor Cluster Stampede2 at TACC**

Intel® Distribution of Caffe* with ImageNet-1K dataset

TRAINING TIME	TRAINING ACCURACY
55 Minutes	75.5% Top-1
LOWER IS BETTER	*HIGHER IS BETTER*

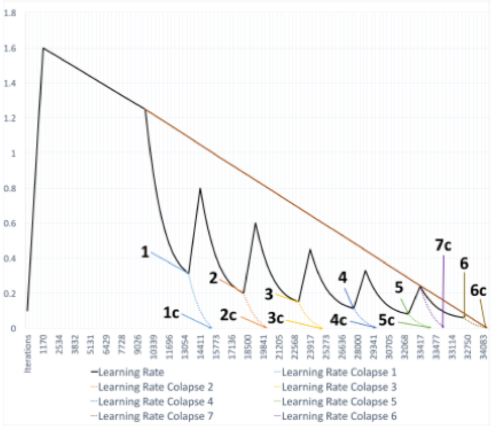
512 Intel® Xeon Phi™ Processor Nodes

Global BS= 10240

***TACC** *(Texas Advanced Computing Center): https://www.tacc.utexas.edu/*

Configuration Details on Slide: Stampede2*/TACC* Configuration Details:
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit: http://www.intel.com/performance.

INCREASING ACCURACY FURTHER USING COLLAPSED ENSEMBLES



No. on plot	Top-1 % acc.	Top-5 % acc.
1	68.33	88.71
1c	75.50	92.83
2	71.54	90.78
2c	76.15	93.17
3	73.28	91.58
3c	76.50	93.24
4	73.31	91.53
4c	76.57	93.24
5	73.89	91.97
5c	76.83	93.32
6	74.49	92.13
6c	76.81	93.32
7c	76.70	93.32

Fig. 3. Plot of learning rate behaviour when obtaining the ensemble snapshots

Collapsed ensembles

Similar in fashion to the learning-rate collapses:

- However, after performing a partial collapse, LR is again increased
- Cycling the LR:
  - Improves single-model accuracy faster
  - Ensemble of the collapsed points leads to 77.5% accuracy using a ResNet-50 regular training budget

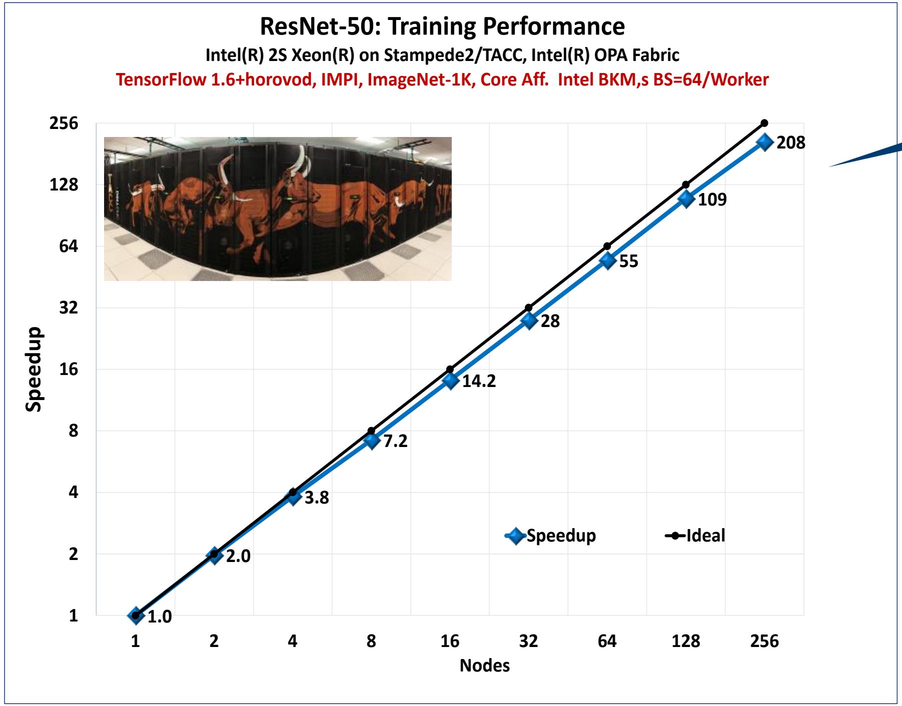
https://github.com/sara-nl/caffe/tree/master/models/intel_optimized_models/multinode/resnet50_custom_lr

PERFORMANCE WITH TENSORFLOW

1. ResNet-50 Benchmark Scaling With TensorFlow

Intel® Xeon® Platinum 8160 processor Cluster Stampede2 at TACC

Joint work with SURFsara/Netherlands,



81% Efficiency with TensorFlow

ResNet-50 with ImageNet-1K on 256 Nodes on Stampede2/TACC:

- Improved single-node perf with multi-workers/node
- 81% scaling efficiency
- Batch size of 64 per worker: Global BS=64K
- 16400 Images/sec on 256 nodes
- 26700 images/sec on 512 nodes
- Time-To-Train: ~2 Hrs on 256 Nodes

2. Monte Carlo → 3D GANs architecture

Joint Work with CERN and SURFsara at ISC18

Problem: Complex physics and geometry modeling via Monte Carlo

Heavy computation requirements

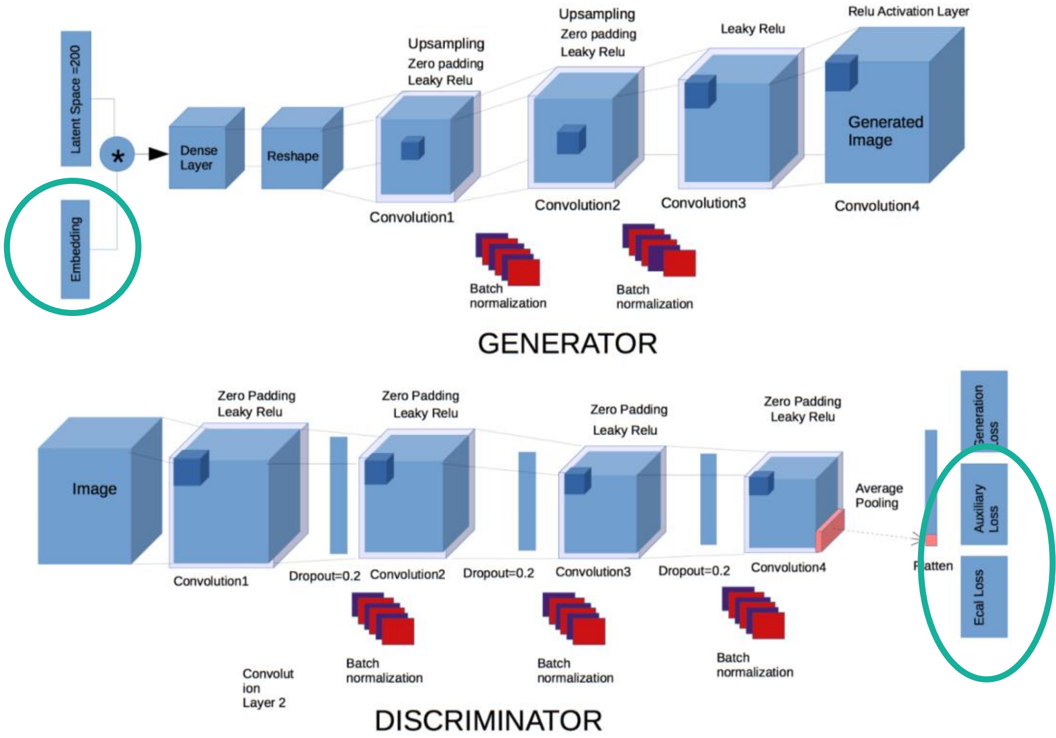
>50% of WLCG power for simulations

Current code cannot cope (HL-LHC in 2025)

Approach: 3D conditional GAN

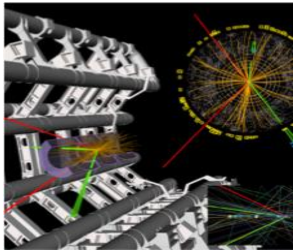
- reproduce full volumes of shower reconstruct in one go
- with two auxiliary regression tasks

Based on 3D convolution/deconvolutions to describe whole volume



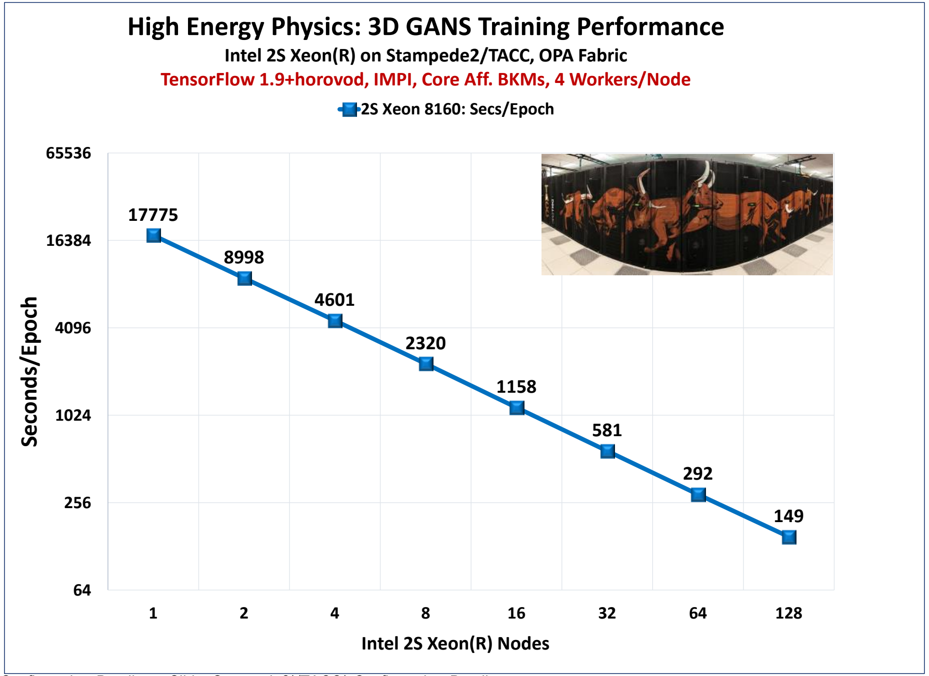
3D-GANs for High Energy Physics/Large Hadron Collider

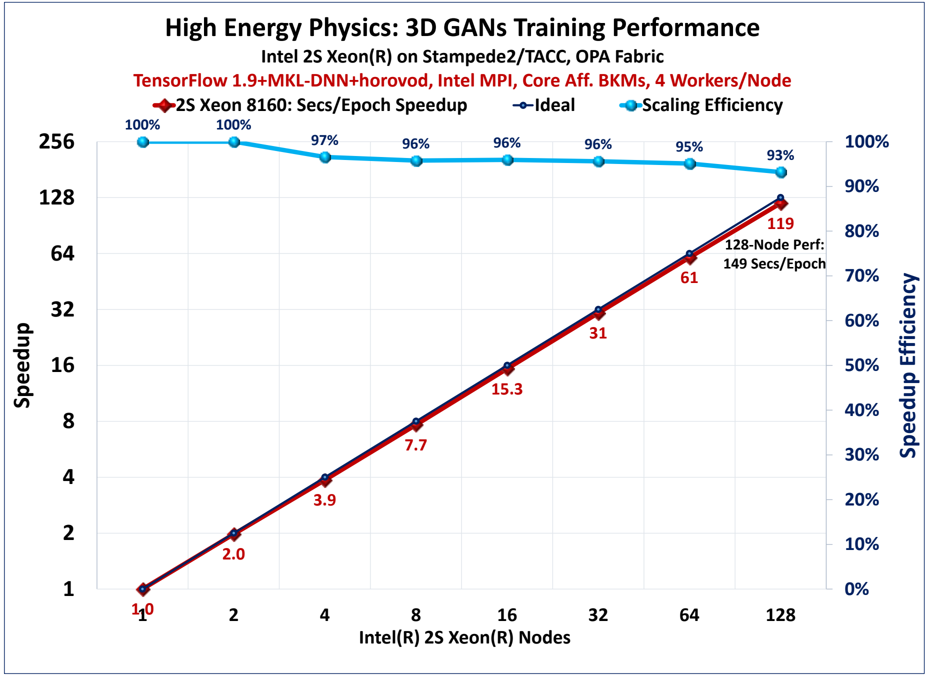
3D GANs instead of Monte Carlo Fast Simulations for detector particles with same accuracy



Joint Work with CERN and SURFsara

93% Scaling Efficiency up to 128 Xeon nodes





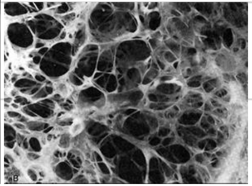
Configuration Details on Slide: Stampede2*/TACC* Configuration Details:

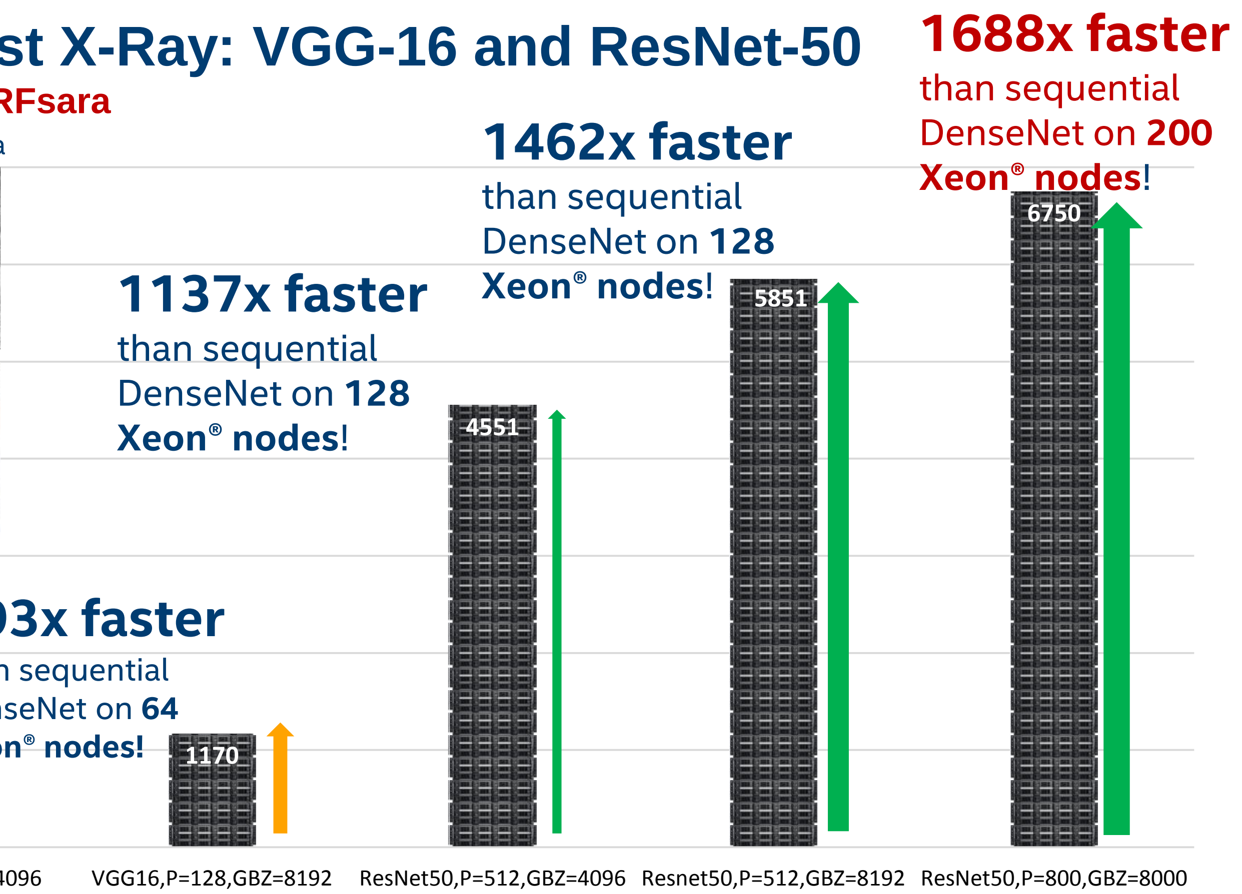
3. AI Radiologist Chest X-Ray: VGG-16 and ResNet-50

Joint work with DellEMC and SURFsara

Chest X-Ray

Severe Emphysema





Dell EMC Poweredge C6420 with dual Intel® Xeon® Scalable Gold 6148 on Intel ® Omni-Path network. ResNet50 tests performed with TensorFlow+Horovod

Summary

- Intel TensorFlow delivers significant performance gains over baseline
- Setting the correct run-time parameters essential for good performance
- Good support for MPI through plugins and scales ~100's of compute nodes
- Good experiences reported by NERSC/LBNL researchers using Intel TensorFlow on KNL's (CORI system)

intel®

Legal Disclaimers

- Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families: Go to: Learn About Intel® Processor Numbers http://www.intel.com/products/processor_number
- Some results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.
- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.
- Intel does not control or audit the design or implementation of third party benchmarks or Web sites referenced in this document. Intel encourages all of its customers to visit the referenced Web sites or others where similar performance benchmarks are reported and confirm whether the referenced benchmarks are accurate and reflect performance of systems available for purchase.
- Relative performance is calculated by assigning a baseline value of 1.0 to one benchmark result, and then dividing the actual benchmark result for the baseline platform into each of the specific benchmark results of each of the other platforms, and assigning them a relative performance number that correlates with the performance improvements reported.
- SPEC, SPECint, SPECfp, SPECrate, SPECpower, SPECjbb, SPECompG, SPEC MPI, and SPECjEnterprise* are trademarks of the Standard Performance Evaluation Corporation. See http://www.spec.org for more information.
- TPC Benchmark, TPC-C, TPC-H, and TPC-E are trademarks of the Transaction Processing Council. See http://www.tpc.org for more information.
- No computer system can provide absolute reliability, availability or serviceability. Requires an Intel® Xeon® processor E7-8800/4800/2800 v2 product families or Intel® Itanium® 9500 series-based system (or follow-on generations of either.) Built-in reliability features available on select Intel® processors may require additional software, hardware, services and/or an internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.
  For systems also featuring Resilient System Technologies: No computer system can provide absolute reliability, availability or serviceability. Requires an Intel® Run Sure Technology-enabled system, including an enabled Intel processor and enabled technology(ies). Built-in reliability features available on select Intel® processors may require additional software, hardware, services and/or an Internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.
  For systems also featuring Resilient Memory Technologies: No computer system can provide absolute reliability, availability or serviceability. Requires an Intel® Run Sure Technology-enabled system, including an enabled Intel® processor and enabled technology(ies). built-in reliability features available on select Intel® processors may require additional software, hardware, services and/or an Internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

System configuration

❑ Single Node Performance: SKX

System configuration:
CPU Thread(s) per core: 2 **Core(s) per socket**: 28 **Socket(s)**: 2 **NUMA node(s)**: 2 **CPU family**: 6 **Model**: 85 **Model name**: Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz Stepping: 4
HyperThreading: ON **Turbo**: ON **Memory** 376GB (12 x 32GB) 24 slots, 12 occupied 2666 MHz Disks Intel RS3WC080 x 3 (800GB, 1.6TB, 6TB) **BIOS** SE5C620.86B.00.01.0004.071220170215 **OS** Centos Linux 7.4.1708 (Core) Kernel 3.10.0-693.11.6.el7.x86_64
TensorFlowSource: https://github.com/tensorflow/tensorflow
TensorFlow Commit ID: 926fc13f7378d14fa7980963c4fe774e5922e336.

Model	Data_format	Intra_op	Inter_op	OMP_NUM_THREADS	KMP_BLOCKTIME
VGG16	NCHW	56	1	56	1
InceptionV3	NCHW	56	2	56	1
ResNet50	NCHW	56	2	56	1

❑ Single Node Performance: KNL

System configuration:
CPU Thread(s) per core: 4 **Core(s) per socket**: 68 **Socket(s)**: 1 **NUMA node(s)**: 2 **CPU family**: 6 **Model**: 87 **Model name**: Intel(R) Xeon(R) Phi ™ CPU 7250 @ 1.4GHz Stepping: 1
HyperThreading: ON
TensorFlowSource: https://github.com/tensorflow/tensorflow v1.2

Benchmark	Data Format	Inter_op	Intra_op	KMP_BLOCKTIME	OMP_NUM_THREADS	Batch size
ConvNet- AlexnetNet	NCHW	1	136	30	136	2048
ConvNet-Googlenet V1	NCHW	2 training 1 inference	68	Infinite	68	256
ConvNet-VGG	NCHW	1	136	1	136	128

Stampede2*/TACC* Configuration Details: Intel® Xeon Phi™

***Stampede2/TACC**: https://portal.tacc.utexas.edu/user-guides/stampede2

Compute Cluster: Intel® Xeon Phi™ processor 7250 (68 Cores, 4 HW Threads per core, 1.4 GHz, 16GB high-speed MCDRAM, 32KB L1 data cache per core; 1MB L2 per two-core tile. In default config, MCDRAM operates as 16GB direct-mapped L3, 96GB DDR4 plus 16GB high-speed MCDRAM, All but 504 KNL nodes have a 132GB /tmp partition on a 200GB Solid State Drive (SSD). Intel® Omni-Path Host Fabric Interface, dual-rail. Software: Intel® MPI Library 2017 Update 4Intel® MPI Library 2019 Technical Preview OFI 1.5.0PSM2 w/ Multi-EP. Red Hat* Enterprise Linux 6.7.

Intel® Distribution of Caffe*: http://github.com/intel/caffe/, revision 8012927bf2bf70231cbc7ff55de0b1bc11de4a69.
Intel® MKL version: mklml_lnx_2018.0.20170425; Intel® MLSL version: l_mlsl_2017.1.016

Model: Topology specs from https://github.com/intel/caffe/tree/master/models/intel_optimized_models (ResNet-50) and modified for wide-RedNet-50.; Batch size as stated in the performance chart

Time-To-Train: measured using "train" command. Data copied to memory on all nodes in the cluster before training. No input image data transferred over the fabric while training; Performance measured for node count: 128, 192, 256, 512, 768 & Performance projected for node count: 1-64.

Performance measured with:
export OMP_NUM_THREADS=64 (the remaining 4 cores are used for driving communication), export I_MPI_FABRICS=tmi, export I_MPI_TMI_PROVIDER=psm2

OMP_NUM_THREADS=64 KMP_AFFINITY="proclist=[0-63],granularity=thread,explicit" KMP_HW_SUBSET=1t MLSL_NUM_SERVERS=4 mpiexec.hydra -PSM2 -l -n $SLURM_JOB_NUM_NODES -ppn 1 -f hosts2 -genv OMP_NUM_THREADS 64 -env KMP_AFFINITY "proclist=[0-63],granularity=thread,explicit" -env KMP_HW_SUBSET 1t -genv I_MPI_FABRICS tmi -genv I_MPI_HYDRA_BRANCH_COUNT $SLURM_JOB_NUM_NODES -genv I_MPI_HYDRA_PMI_CONNECT alltoall sh -c 'cat /ilsvrc12_train_lmdb_striped_64/data.mdb > /dev/null ; cat /ilsvrc12_val_lmdb_striped_64/data.mdb > /dev/null ; ulimit -u 8192 ; ulimit -a ; numactl -H ; /caffe/build/tools/caffe train --solver=/caffe/models/intel_optimized_models/multinode/resnet_50_256_nodes_8k_batch/solver_poly_quick_large.prototxt -engine "MKL2017"

VLAB at Intel® Configuration Details: Intel® Xeon Phi™

***VLAB/Intel® Internal Cluster**:

Compute Cluster: Intel® Xeon Phi™ processor 7250 (68 Cores, 4 HW Threads per core, 1.4 GHz, 16GB high-speed MCDRAM, 32KB L1 data cache per core; 1MB L2 per two-core tile. In default config, MCDRAM operates as 16GB direct-mapped L3, 192GB DDR4, Intel® Omni-Path Host Fabric Interface, dual-rail. Software: Intel® MPI Library 2017 Update 4Intel® MPI Library 2019 Technical Preview OFI 1.5.0PSM2 w/ Multi-EP, Red Hat* Enterprise Linux 6.7,

Intel® Distribution of Caffe*: http://github.com/intel/caffe/), revision f96b759f71b2281835f690af267158b82b150b5c.
Intel MKL version: mklml_lnx_2018.0.20170425; Intel MLSL version: l_mlsl_2017.1.016

Model: https://github.com/intel/caffe/tree/master/models/intel_optimized_models (ResNet-50) and modified for Wide-ResNet-50. Batch size as stated in the performance chart

Time-To-Train: measured using "train" command. Data copied to memory on all nodes in the cluster before training. No input image data transferred over the fabric while training; Performance measured for node count: 200, 210, 240.

Performance measured with:
export OMP_NUM_THREADS=64 (the remaining 4 cores are used for driving communication), export I_MPI_FABRICS=tmi, export I_MPI_TMI_PROVIDER=psm2

OMP_NUM_THREADS=64 KMP_AFFINITY="proclist=[0-63],granularity=thread,explicit" KMP_HW_SUBSET=1t MLSL_NUM_SERVERS=4 mpiexec.hydra -PSM2 -l -n $SLURM_JOB_NUM_NODES -ppn 1 -f hosts2 -genv OMP_NUM_THREADS 64 -env KMP_AFFINITY "proclist=[0-63],granularity=thread,explicit" -env KMP_HW_SUBSET 1t -genv I_MPI_FABRICS tmi -genv I_MPI_HYDRA_BRANCH_COUNT $SLURM_JOB_NUM_NODES -genv I_MPI_HYDRA_PMI_CONNECT alltoall sh -c 'cat /ilsvrc12_train_lmdb_striped_64/data.mdb > /dev/null ; cat /ilsvrc12_val_lmdb_striped_64/data.mdb > /dev/null ; ulimit -u 8192 ; ulimit -a ; numactl -H ; /caffe/build/tools/caffe train --solver=/caffe/models/intel_optimized_models/multinode/resnet_50_256_nodes_8k_batch/solver_poly_quick_large.prototxt -engine "MKL2017"

Stampede2*/TACC* Configuration Details: 2S Intel® Xeon®

***Stampede2/TACC**: https://portal.tacc.utexas.edu/user-guides/stampede2

Compute Nodes: 2 sockets Intel® Xeon® Platinum 8160 CPU with 24 cores each @ 2.10GHz for a total of 48 cores per node, 2 Threads per core, L1d 32K; L1i cache 32K; L2 cache 1024K; L3 cache 33792K, 96 GB of DDR4, Intel® Omni-Path Host Fabric Interface, dual-rail. Software: Intel® MPI Library 2017 Update 4Intel® MPI Library 2019 Technical Preview OFI 1.5.0PSM2 w/ Multi-EP, 10 Gbit Ethernet, 200 GB local SSD, Red Hat* Enterprise Linux 6.7.

TensorFlow 1.6: Built & Installed from source: https://www.tensorflow.org/install/install_sources

Model: Topology specs from https://github.com/tensorflow/tpu/tree/master/models/official/resnet (ResNet-50); Batch size as stated in the performance chart

Convergence & Performance Model: https://surfdrive.surf.nl/files/index.php/s/xrEFLPvo7IDRARs

Dataset: ImageNet2012-1K: http://www.image-net.org/challenges/LSVRC/2012/

Performance measured on 256 Nodes with:
OMP_NUM_THREADS=24 HOROVOD_FUSION_THRESHOLD=134217728 export I_MPI_FABRICS=tmi, export I_MPI_TMI_PROVIDER=psm2 \
mpirun -np 512 -ppn 2 python resnet_main.py --train_batch_size 8192 --train_steps 14075 --num_intra_threads 24 --num_inter_threads 2 --mkl=True --data_dir=/scratch/04611/valeriuc/tf-1.6/tpu_rec/train --model_dir model_batch_8k_90ep --use_tpu=False --kmp_blocktime 1

DellEMC Zenith Cluster Configuration Details

Compute Nodes: 2 sockets Intel® Xeon® Platinum 8160 CPU with 24 cores each @ 2.10GHz for a total of 48 cores per node, 2 Threads per core, L1d 32K; L1i cache 32K; L2 cache 1024K; L3 cache 33792K, 96 GB of DDR4, Intel® Omni-Path Host Fabric Interface, dual-rail. Software: Intel® MPI Library 2017 Update 4Intel® MPI Library 2019 Technical Preview OFI 1.5.0PSM2 w/ Multi-EP, 10 Gbit Ethernet, 200 GB local SSD, Red Hat* Enterprise Linux 6.7.

TensorFlow 1.6: Built & Installed from source: https://www.tensorflow.org/install/install_sources

ResNet-50 Model: Topology specs from https://github.com/tensorflow/tpu/tree/master/models/official/resnet
DenseNet-121Model: Topology specs from https://github.com/liuzhuang13/DenseNet

Convergence & Performance Model: https://surfdrive.surf.nl/files/index.php/s/xrEFLPvo7IDRARs

Dataset:
ImageNet2012-1K: http://www.image-net.org/challenges/LSVRC/2012/
ChexNet: https://stanfordmlgroup.github.io/projects/chexnet/

Performance measured with:

OMP_NUM_THREADS=24 HOROVOD_FUSION_THRESHOLD=134217728 export I_MPI_FABRICS=tmi, export I_MPI_TMI_PROVIDER=psm2 \
mpirun -np 512 -ppn 2 python resnet_main.py --train_batch_size 8192 --train_steps 14075 --num_intra_threads 24 --num_inter_threads 2 --
mkl=True --data_dir=/scratch/04611/valeriuc/tf-1.6/tpu_rec/train --model_dir model_batch_8k_90ep --use_tpu=False --kmp_blocktime 1