Whitepaper

Deep Learning Performance Scale-out

Revision: **1.2**
Issue Date: **3/16/2020**

Abstract

In this whitepaper we evaluated the training performance of Scale-out implementation with the latest software stack and compared it with the results obtained in our previous paper [0]. Using TensorFlow as the primary framework and tensorflow benchmark models, the performance was compared in terms of throughput images/sec on ImageNet dataset, at a single node and multi-node level. We tested some of the more popular neural networks architectures for this comparison and demonstrated the scalability capacity of Dell EMC PowerEdge Servers powered by NVIDIA V100 Tensor Core GPUs.

Revisions

Date	Description
3/16/2020	Initial release

Acknowledgements

This paper was produced by the following:

Name	
Vilmara Sanchez	Dell EMC - Software Engineer
Bhavesh Patel	Dell EMC - Distinguished Engineer
Josh Anderson	Dell EMC - System Engineer (contributor)

We would like to acknowledge:

- Technical Support Team - Mellanox Technologies
- Uber Horovod GitHub Team
- Nvidia Support team

Table of Contents

Motivation

With the recent advances in the field of Machine Learning and especially Deep Learning, it's becoming more and more important to figure out the right set of tools that will meet some of the performance characteristics for these workloads. Since Deep Learning is compute intensive, the use of accelerators like GPU become the norm, but GPUs are premium components and often it comes down to what is the performance difference between a system with and without GPU. In that sense Dell EMC is constantly looking to support the business goals of customers by building highly scalable and reliable infrastructure for Machine Learning/Deep Learning workloads and exploring new solutions for large scale distributed training to optimize the return on investment (ROI) and Total Cost of Ownership (TCO).

Test Methodology

We have classified TF benchmark tests in two categories: short and long tests. During the development of the short tests, we experimented with several configurations to determine the one that yielded the highest throughput in terms of images/second, then we selected that configuration to run the long tests to reach certain accuracy targets.

Short Tests

The tests consisted of 10 warmup steps and then another 100 steps which were averaged to get the actual throughput. The benchmarks were run with 1 NVIDIA GPU to establish a baseline number of images/sec and then increasing the number of GPUs to 4 and 8. These tests allow us to experiment with the parameter tuning of the models in distributed mode.

Long Tests

The tests were run using 90 epochs as the standard for ResNet50. This criterion was used to determine the total training time on C4140-M servers in distributed mode with the best parameter tuning found in the short tests and using the maximum number of GPUs supported by the system.

In the section below, we describe the setup used, and *Table 1* gives an overall view on the test configuration.

Testing Setup

Commercial Application	Computer Vision - Image classification
Benchmarks code	▪ TensorFlow Benchmarks scripts
Topology	▪ Single Node and Multi Node over InfiniBand
Server	▪ PowerEdge C4140-M (4xV100-16GB-SXM2)
Frameworks	▪ TensorFlow with Horovod library for Distributed Mode[1]
Models	▪ Convolutional neural networks: Inception-v4, vgg19, vgg16, Inception-v3, ResNet-50 and GoogLeNet
Batch size	▪ 128-256

GPU's	▪ 1-8
Performance Metrics	▪ Throughput images/second ▪ Training to convergence at 76.2% TOP-1 ACCURACY
Dataset	▪ ILSVRC2012 - ImageNet
Environment	▪ Docker

Table 1: Benchmark Setup

Software Stack

The *Table* shows the software stack configuration used to build the environment to run the tests shown in paper [0] and the current tests

Software Stack	Previous Tests	Current Tests
Test Date	February 2019	January 2020
OS	Ubuntu 16.04.4 LTS	Ubuntu 18.04.3 LTS
Kernel	GNU/Linux 4.4.0-128-generic x86_64	GNU/Linux 4.15.0-69-generic x86_64
nvidia driver	396.26	440.33.01
CUDA	9.1.85	10.0
cuDNN	7.1.3	7.6.5
NCCL	2.2.15	2.5.6
TensorFlow	1.10	1.14
Horovod	0.15.2	0.19.0
Python	2.7	2.7
Open MPI	3.0.1	4.0.0
Mellanox OFED	4.3-1	4.7-3
GPUDirect RDMA	1.0-7	1.0-8
Single Node - Docker Container	TensorFlow/tensorflow:nightly-gpu-py3	nvidia/cuda:10.0-devel-ubuntu18.04
Multi Node - Docker Container built from	nvidia/cuda:9.1-devel-ubuntu16.04	nvidia/cuda:10.0-devel-ubuntu18.04
Benchmark scripts	tf_cnn_benchmarks	tf_cnn_benchmarks

Table 2: OS & Driver Configurations

Distributed Setup

The tests were run in a docker environment. **Error! Reference source not found.** 1 below shows the different logical layers involved in the software stack configuration. Each server is connected to the InfiniBand switch; has installed on the Host the Mellanox OFED for Ubuntu, the Docker CE, and the GPUDirect RDMA API; and the container image that was built with Horovod and Mellanox OFED among other supporting libraries. To build the extended container image, we used the Horovod docker file and modified it by adding the installation for Mellanox OFED drivers [2]. It was built from nvidia/cuda:10.0-devel-ubuntu18.04

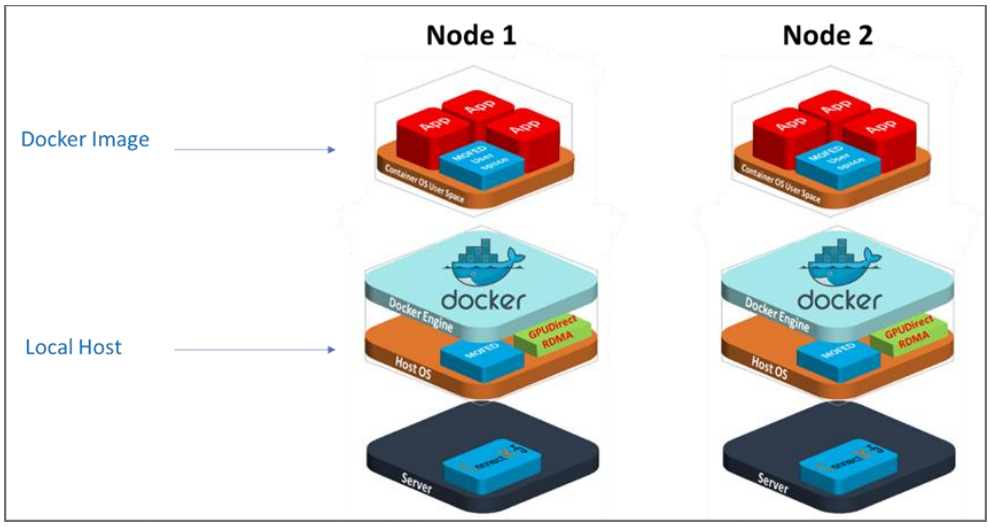

Figure 1: Servers Logical Design. Source: Image adapted from https://community.mellanox.com/docs/DOC-2971

Error! Reference source not found. *2* below shows how PowerEdge C4140-M is connected via InifniBand fabric for multi-node testing.

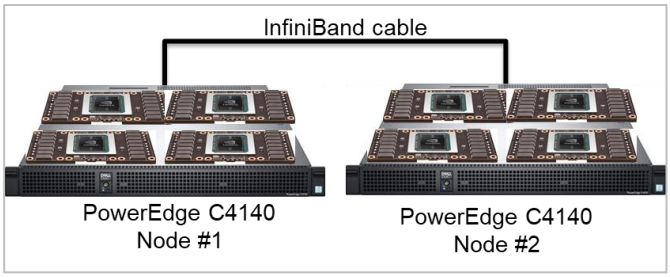

Figure 2: Using Mellanox CX5 InfiniBand adapter to connect PowerEdge C4140 in multi-node configuration

PowerEdge C4140-M Details

The Dell EMC PowerEdge C4140, an accelerator-optimized, high density 1U rack server, is used as the compute node unit in this solution. The PowerEdge C4140 can support four NVIDIA V100 Tensor Core GPUs, both the V100-SXM2 (with high speed NVIDIA NVLink interconnect) as well as the V100-PCIe models.

Figure 3: PowerEdge C4140 Server

The Dell EMC PowerEdge C4140 supports NVIDIA V100 with NVLink in topology 'M' with a high bandwidth host to GPU communication is one of the most advantageous topologies for deep learning. Most of the competitive systems, supporting either a 4-way, 8-way or 16-way NVIDIA

Volta SXM, use PCIe bridges and this limits the total available bandwidth between CPU to GPU. See *Table 2* with the Host-GPU Complex PCIe Bandwidth Summary.

Configuration	Link Interface b/n CPU-GPU complex	Total Bandwidth	Notes
M	4x16 Gen3	128GB/s	Each GPU has individual x16 Gen3 to Host CPU

Table 2: Host-GPU Complex PCIe Bandwidth Summary

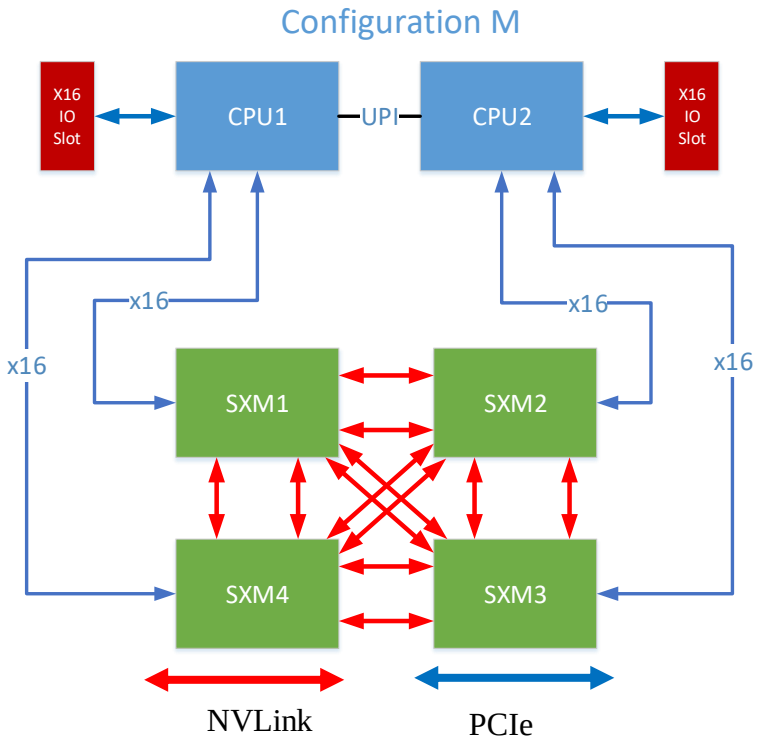

Figure 4: C4140 Configuration-M

Performance Results - Short Tests for Parameter Tuning

Below are the results for the short tests using TF 1.14. In this section we tested all the models in multi node mode and compared the results obtained with TF 1.10 in 2019.

Throughput CNN Models TF 1.10 vs TF 1.14

The *Figure 5* several CNN models comparing results with TF 1.10 vs TF 1.14. In *Figure 6* we notice that the performance gain is about 1.08X (or 8%) between the two releases.

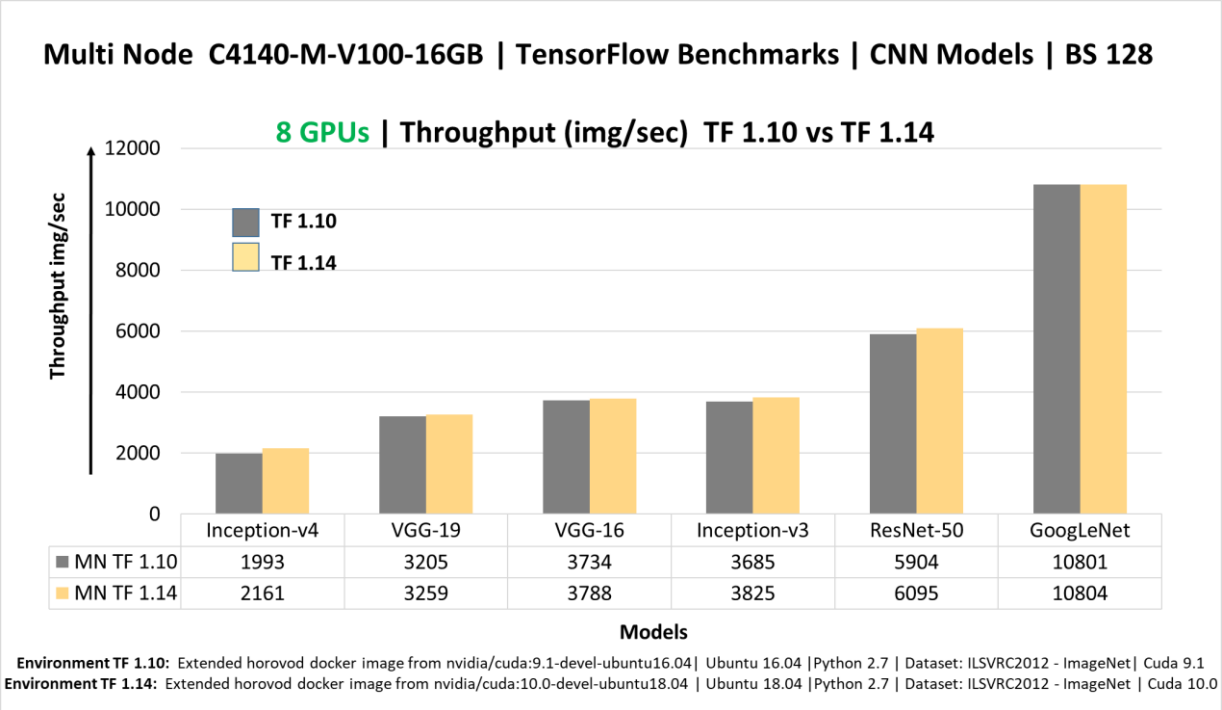

Figure 5: Multi Node PowerEdge C4140-M – Several CNN Models TF 1.10 vs TF 1.14

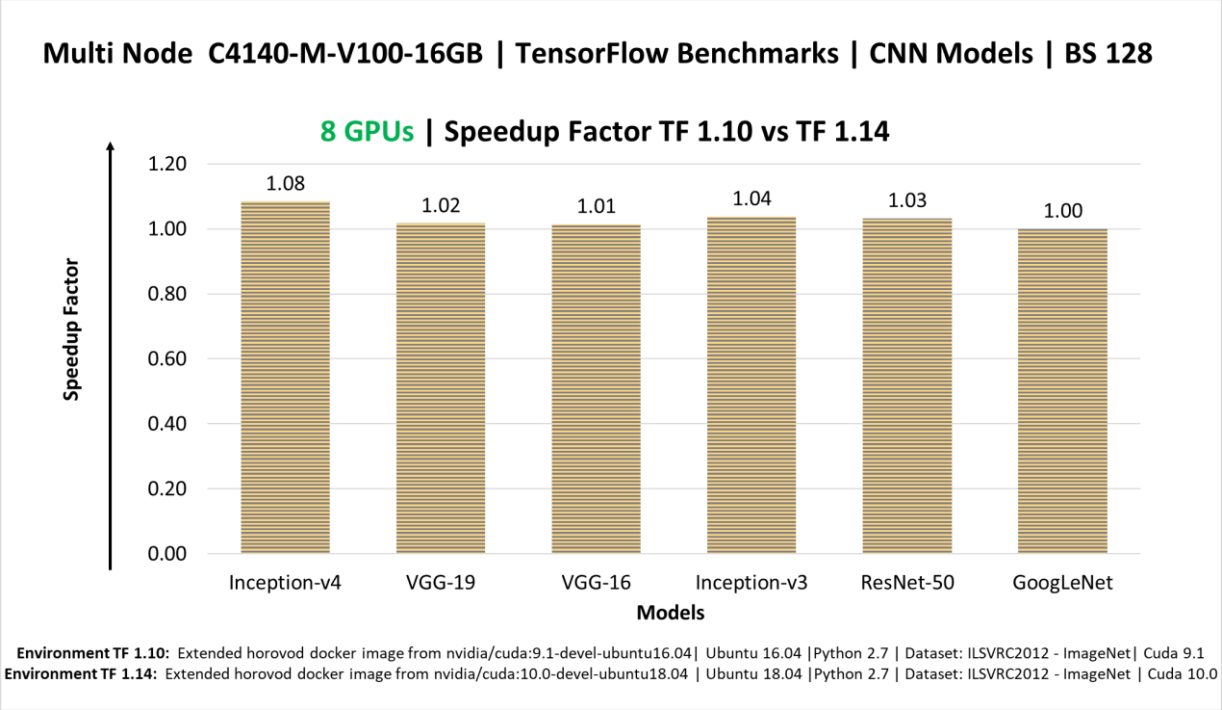

Figure 6 Multi Node PowerEdge C4140-M – Several CNN Models TF 1.10 vs TF 1.14 (Speedup factor)

Performance Gain with XLA

Since there was not much performance gain with the basic configuration, we decided to explore the limits of GPU performance using other parameters. We looked at XLA (Accelerated Linear Algebra) [3], by adding the flag –xla=true at the script level. By default, the TensorFlow graph executor "executes" the operations with individual kernels, one kernel for the multiplication, one kernel for the addition and one for the reduction. With XLA, these operations are "fused" in just one kernel; keeping the intermediate and final results in the GPU, reducing memory operations, and therefore improving performance.

See below *Figure 7* for results and *Figure 8* for speedup factors across the models. The models inception-v4, inception-v3 and ResNet-50 showed much better performance using XLA, with speedup factors from 1.35X up to 1.43X. Since the ResNet-50 model is most widely used, we used it to continue the rest of the tests.

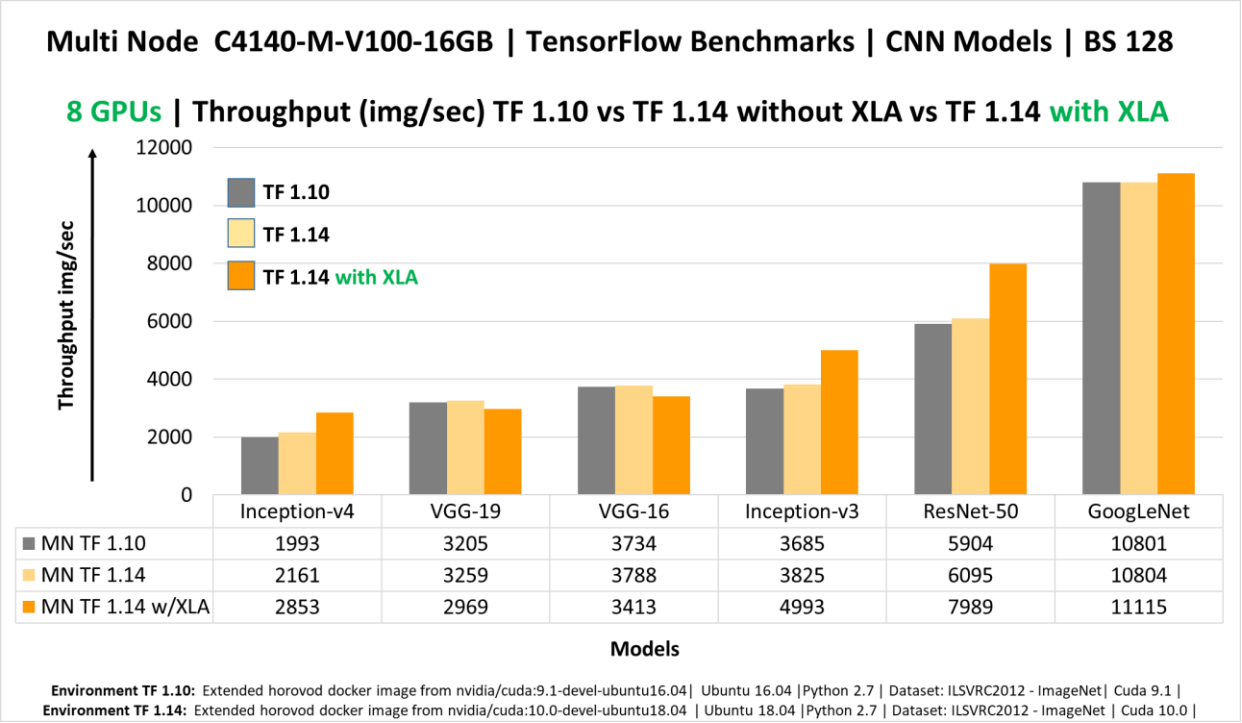

Figure 7: Multi Node PowerEdge C4140-M. Several CNN Models TF 1.10 vs TF 1.14 + XLA

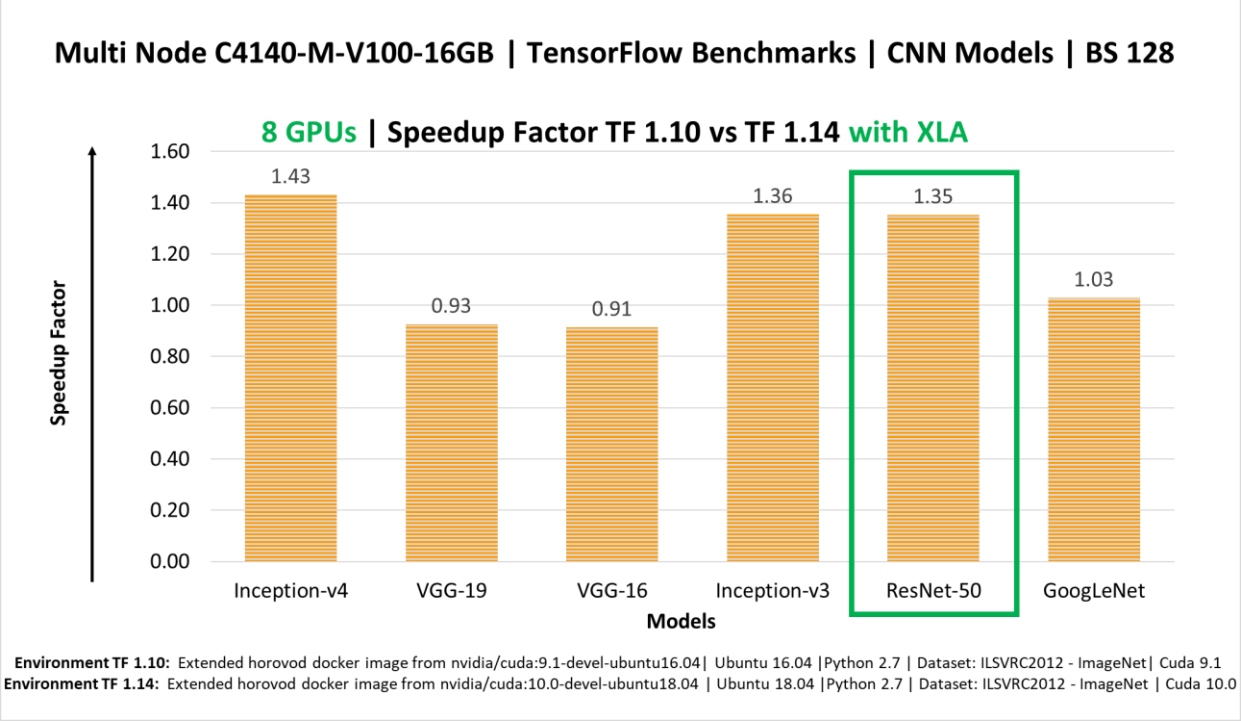

Figure 8: Multi Node PowerEdge C4140-M. Several CNN Models TF 1.10 vs TF 1.14 + XLA (Speedup factor)

ResNet-50's Performance with TF 1.14 + XLA

In this section, we evaluated the performance of ResNet-50 model trained with TF 1.14 and TF 1.14 with XLA enabled. The tests were run with 1 GPU, 4 GPUs, and 8 GPUs and the results were compared with those obtained for version TF 1.10 from our previous paper [0]. Also, we explored the performance using batch size of 128 and 256. See *Figure 9* and *Figure 10*.

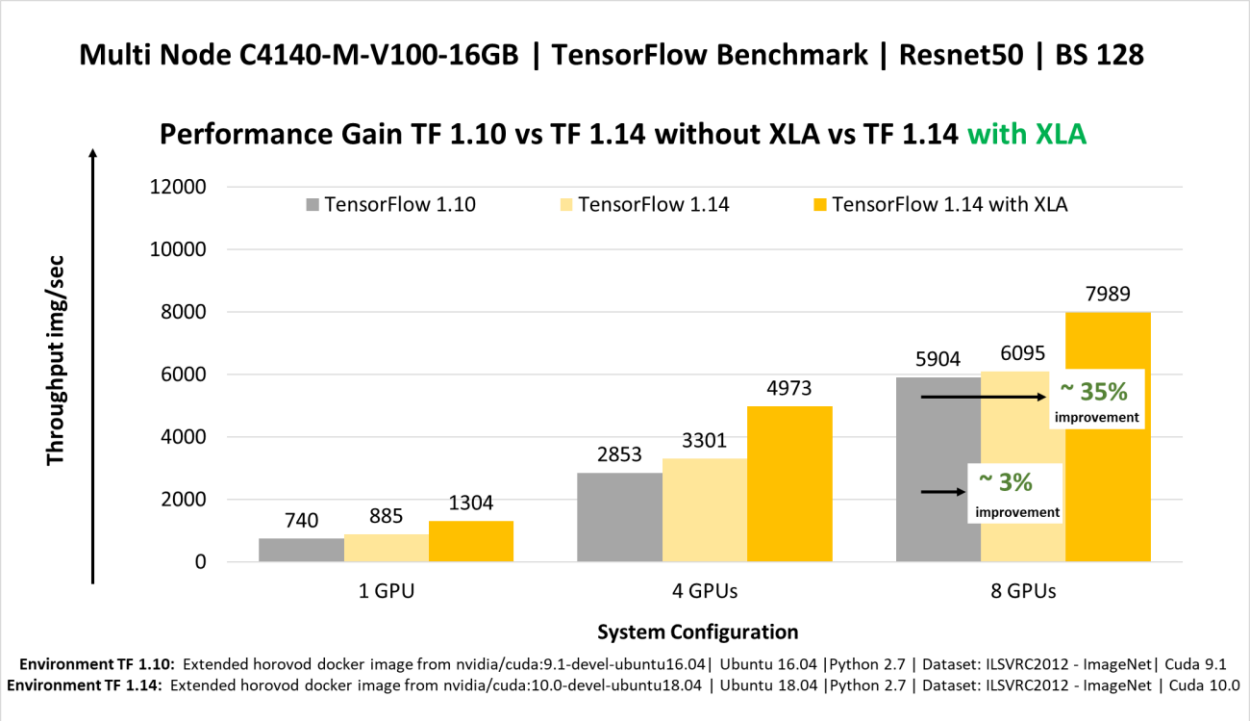

Figure 9: Multi Node PowerEdge C4140-M. ResNet-50 BS 128 TF 1.10 vs TF 1.14 vs TF 1.14 + XLA

As we saw in the previous section ResNet-50 with batch size 128 with 8 GPUs had a performance gain of ~3% with TF 1.10 vs TF 1.14, and ~35% of performance gain with TF 1.10 vs TF 1.14 with XLA enabled, see *Figure 9*. On the other hand, ResNet-50 with batch size 256 with 8 GPUs had a performance gain of ~2% with TF 1.10 vs TF 1.14, and ~46% of performance gain with TF 1.10 vs TF 1.14 with XLA enabled, see *Figure 10.* Due to the higher performance of ResNet-50 with batch size 256, we have selected it to further optimize performance.

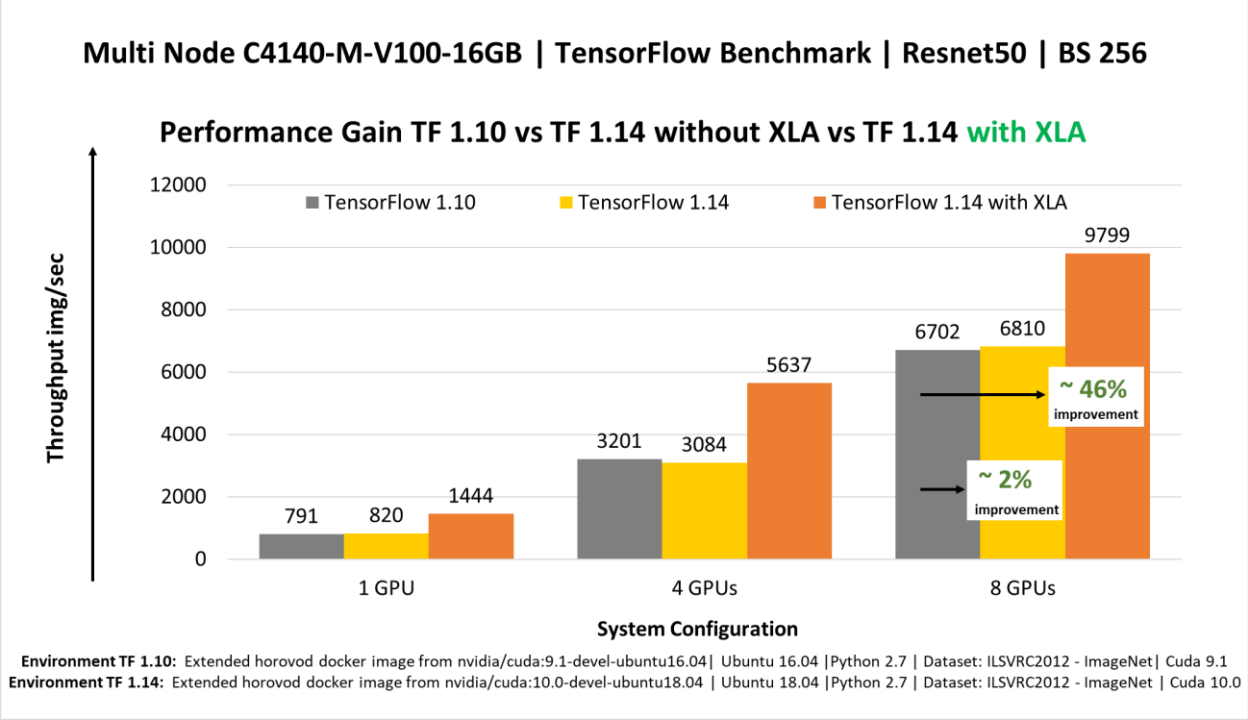

Figure 10: Multi Node PowerEdge C4140-M. ResNet-50 BS 256 TF 1.10 vs TF 1.14 vs TF 1.14 + XLA

ResNet-50 with TF 1.14 + XLA + GPUDirect RDMA

Another feature explored in our previous paper was GPUDirect RDMA which provides a direct P2P (Peer-to-Peer) data path between GPU memory using a Mellanox HCA device between the nodes. In this test, we enabled it by adding the NCCL flag `– x NCCL_NET_GDR_LEVEL=3` at the script level (this variable replaced the variable NCCL_IB_CUDA_SUPPORT in NCCL v 2.4.0). `NCCL_NET_GDR_LEVEL` variable allows you to control when to use GPUDirect RDMA between a NIC and a GPU. Example level 3 indicates to use GPUDirect RDMA when GPU and NIC are on the same PCI root complex [4].

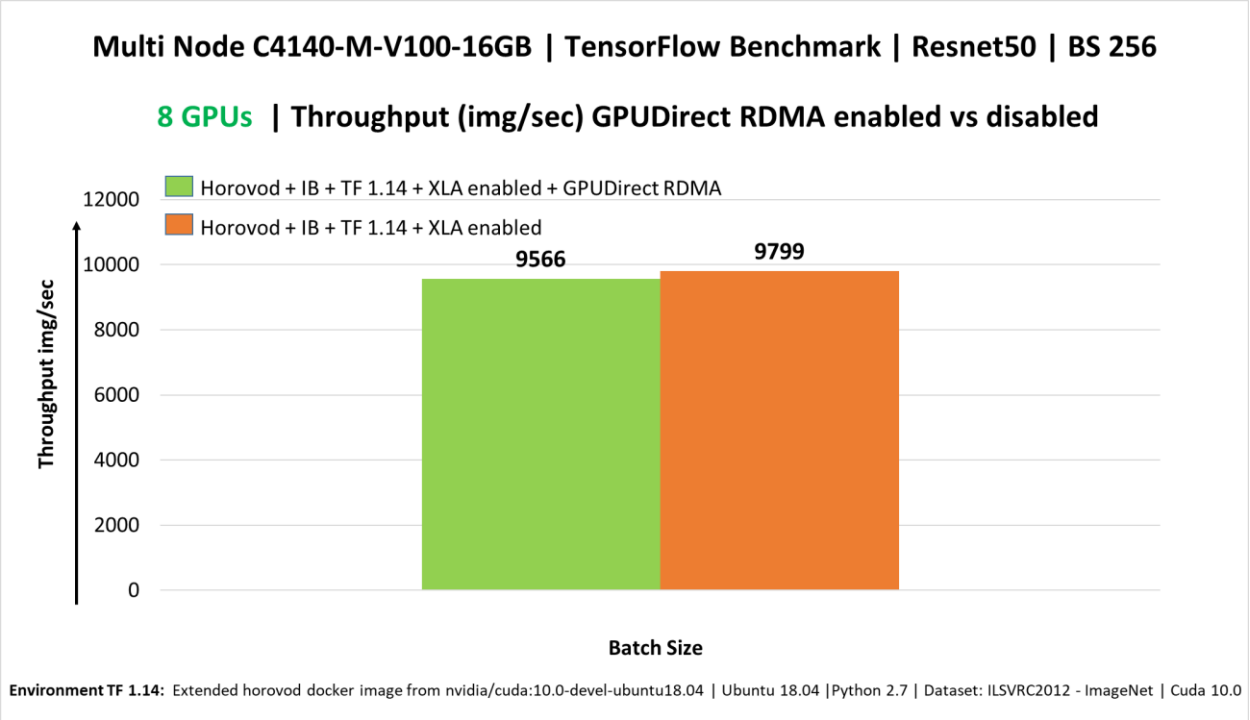

Figure 11: Multi Node PowerEdge C4140-M. ResNet-50 with TF 1.14 + XLA + GPUDirect RDMA

Figure 11 shows the results of ResNet-50 with TF 1.14 w/XLA enabled, with and without GPUDirect RDMA. We did not observe much performance gains using GPUDirect RDMA across nodes i.e. the performance remained the same and hence we did not explore it further in our testing. This is not to say that GPUDirect RDMA does not help when using scale-out, all we are saying is we did not see the performance gains; hence we are not exploring it further in this paper.

ResNet-50's configuration for better performance

Figure 12 summarizes the results of different configurations explored for the short tests and based on the tests we found that the best performance was achieved using the combination below:

ResNet-50 + BS 256 + TF 1.14 + XLA enabled

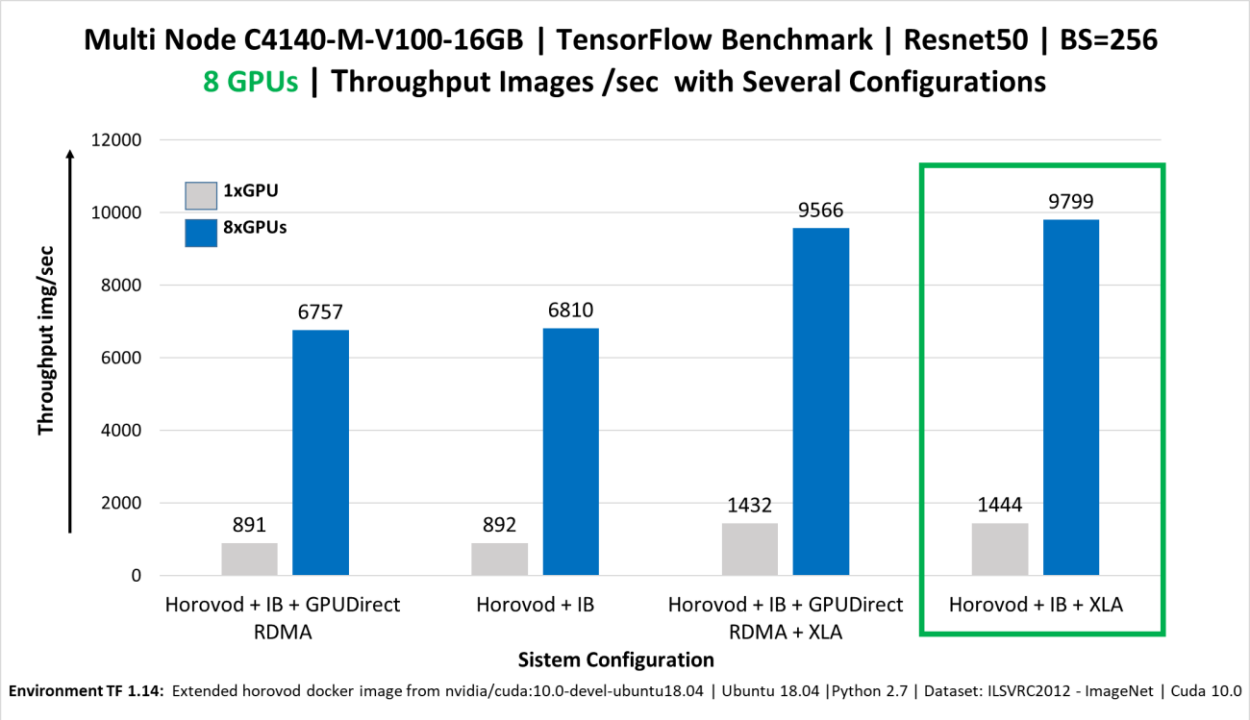

Figure 12: Multi Node PowerEdge C4140-M - ResNet-50's Configuration for Best Performance

ResNet-50's Scale-out

PowerEdge C4140 using Nvidia 4x NVLink architecture scales relatively well when using Uber Horovod distributed training library and Mellanox InfiniBand as the high-speed link between nodes. It scales ~3.9x times within the node and ~6.9x using scale-out for ResNet-50 with batch size 256. See *Figure 13.*

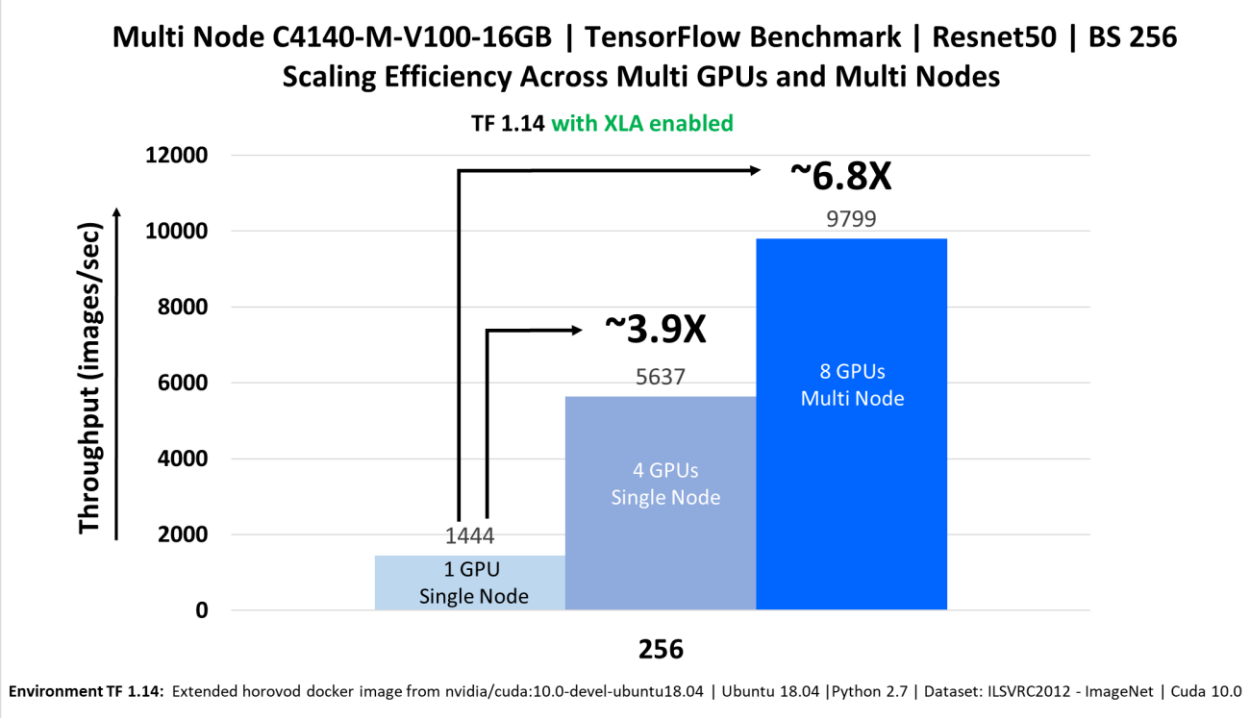

Figure 13: Multi Node PowerEdge C4140-M - ResNet-50's Scale-out vs Scale-up

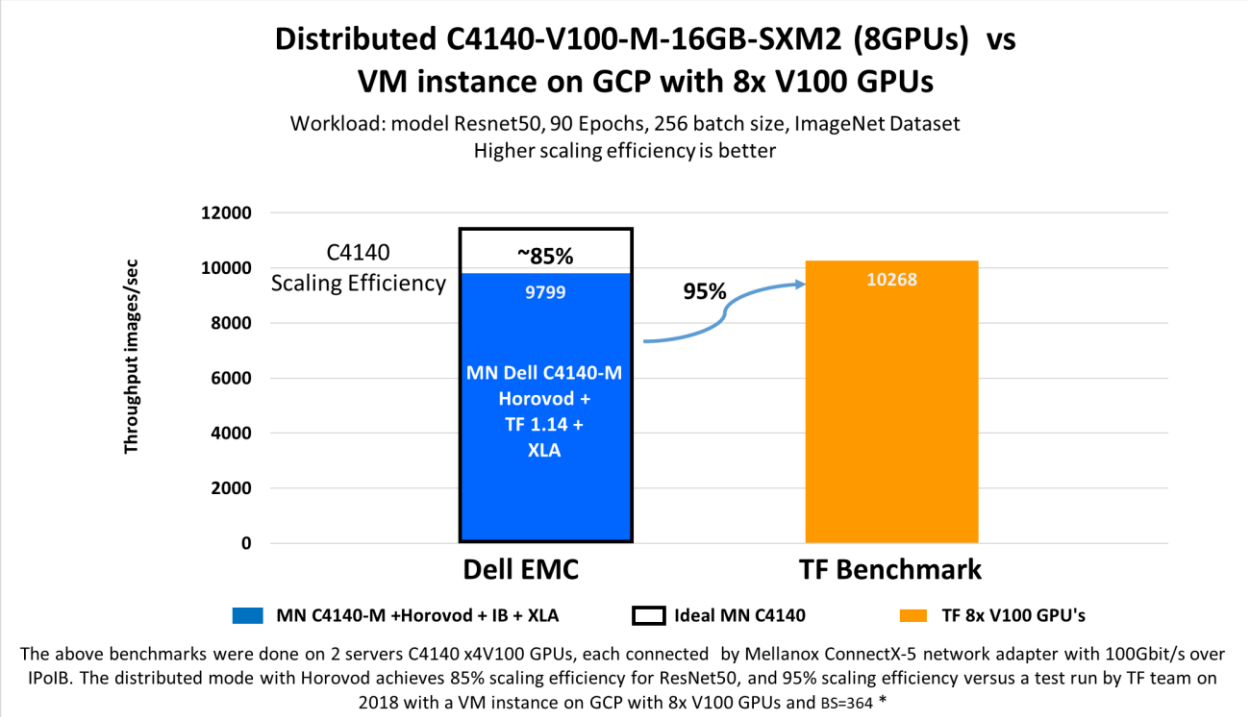

Figure 14: Multi Node PowerEdge C4140-M vs Competitor

The above benchmarks shown in *Figure 14* are done on 2 servers C4140 x4 V100 GPUs, each connected by Mellanox ConnectX-5 network adapter with 100Gbit/s over IPoIB. The Dell EMC distributed mode with Horovod achieved 85% of scaling efficiency for ResNet-50 batch size 256 compared with the ideal performance; on the other hand, it achieved 95% of scaling efficiency versus a test run by TF team on 2018 with a VM (virtual machine) instance on GCP(Google cloud) with 8x V100 GPUs and batch size=364 [5].

Performance Results - Long Tests Accuracy Convergence

Our final tests were to determine the total training time for accuracy convergence with the latest tensorflow version.
In this section we decided to include all the batch sizes we tested in our previous paper and compared it with ResNet-50 using batch size 256.
Figure 15 shows the total training time achieved when running ResNet-50 with different batch sizes and both versions of tensorflow (TF 1.10 vs TF 1.14 with XLA enabled).
On average using TF 1.14 +XLA was ~1.3X faster than our previous tests.

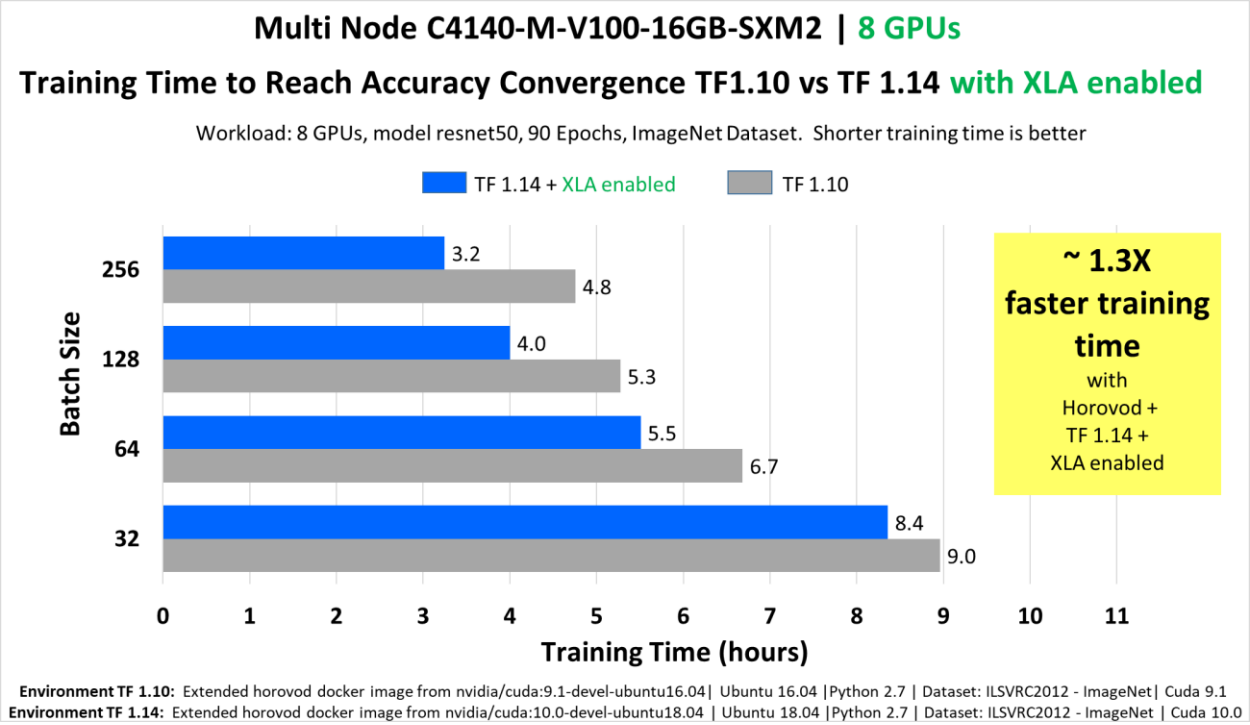

Figure 15: Multi Node PowerEdge C4140-M - ResNet-50's Long Training for Accuracy Conv.

Conclusion

- The performance with TF 1.14 among the models was just slightly superior (~1%-8%) versus TF 1.10. On the other hand, TF 1.14 with XLA boosted the performance up to ~46% among the models ResNet-50, Inception-v3 and Inception-v4.

- In the case of ResNet-50 model, its performance improved up to ~ 3% with TF 1.14, and up to ~46% with TF 1.14 and XLA enabled. ResNet-50 batch size 256 scaled better (1.46X) versus ResNet-50 BS 128 (1.35X).

- The configuration with the highest throughput (img/sec) was ResNet-50 batch size 256 trained with distributed Horovod + TF 1.14 + XLA enabled.

- Dell EMC PowerEdge C4140 using Nvidia 4x NVLink architecture scales relatively well when using Uber Horovod distributed training library and Mellanox InfiniBand as the high-speed link between nodes. It scale-out ~3.9X times within the node and ~6.9X across nodes for ResNet-50 BS 256.

- On average, the training time for the long tests to reach accuracy convergence were ~ 1.3 X faster using distributed Horovod + TF 1.14 + XLA enabled.

- There is a lot of performance improvement being added continuously either at the GPU level, library level or framework level. We are continuously looking at how we can improve our performance results by experimenting with different hyper parameters.

- TensorFlow in multi-GPU/multi-node with Horovod Distributed and XLA support improve model performance and reduce the training time, allowing customers to do more with no additional hardware investment.

Server Features

Server		C4140_V100-16GB-SXM2
CPU		
	CPU model	Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz
GPU		
	GPU model	Tesla V100-SXM2-16GB
	Attached GPUs	4
Features per GPU		
	Driver Version	396.26
	Compute Capability	7
Multiprocessor		
	Multiprocessors (MP)	80
	CUDA Cores/MP	64
	CUDA Cores	5120
	Clock Rate (GHz)	1.53
Memory		
	Global Memory Bandwidth (GB/s)	898
	Global Memory Size (GB)	16
	Constant Memory Size (KB)	64
	L2 Cache Size (MB)	6
	Memcpy Engines	5
Bus Interface PCIe		
	Generation	3
	Link Rate (GB/s)	8
	Link Width	16
Peak Performance	**Floating Point Operations (FLOP)**	
	Half Precision - FP16 (TeraFLOP/s)	31.3
	Single-Precision - FP32 (TeraFLOP/s)	15.7
	Double Precision - FP64 (TeraFLOP/s)	7.8
Power		
	Min Power Limit (W)	100
	Max Power Limit (W)	300

Citation

```
@article {sergeev2018horovod,
  Author = {Alexander Sergeev and Mike Del Balso},
  Journal = {arXiv preprint arXiv: 1802.05799},
  Title = {Horovod: fast and easy distributed deep learning in {TensorFlow}},
  Year = {2018}
}
```

References

- [0] https://downloads.dell.com/manuals/all-products/esuprt_solutions_int/esuprt_solutions_int_solutions_resources/servers-solution-resources_white-papers52_en-us.pdf
- [1] Horovod GitHub, “Horovod Distributed Deep Learning Training Framework” [Online]. Available: https://github.com/horovod/horovod
- [2] Mellanox Community, “How to Create a Docker Container with RDMA Accelerated Applications Over 100Gb InfiniBand Network” [Online]. Available: https://community.mellanox.com/docs/DOC-2971
- [3] TensorFlow, “XLA: Optimizing Compiler for Machine Learning” [Online] Available: https://www.tensorflow.org/xla
- [4] NCCL 2.5, “NCCL Environment Variables” [Online]. Available: https://docs.nvidia.com/deeplearning/sdk/nccl-developer-guide/docs/env.html#nccl-ib-cuda-support
- [5] TensorFlow, “Pushing the limits of GPU performance with XLA” [Online]. Available: https://medium.com/tensorflow/pushing-the-limits-of-gpu-performance-with-xla-53559db8e473

Appendix: Reproducibility

The section below walks through the setting requirements for the distributed Dell EMC system and execution of the benchmarks. Do this for both servers:

- Update Kernel on Linux
- Install Kernel Headers on Linux
- Install Mellanox OFED at local host
- Setup Password less SSH
- Configure the IP over InfiniBand (IPoIB)
- Install CUDA with NVIDIA driver
- install CUDA Toolkit
- Download and install GPUDirect RDMA at the localhost
- Check GPUDirect kernel module is properly loaded
- Install Docker CE and nvidia runtime
- Build - Horovod in Docker with MLNX OFED support
- Check the configuration status on each server (nvidia-smi topo -m && ifconfig && ibstat && ibv_devinfo -v && ofed_info -s)
- Pull the benchmark directory into the localhost
- Mount the NFS drive with the ImageNet data

Run the system as:

On Secondary node (run this first):

```
$ sudo docker run --gpus all -it --network=host -v /root/.ssh:/root/.ssh --cap-
add=IPC_LOCK -v /home/dell/imagenet_tfrecords/:/data/ -v
/home/dell/benchmarks/:/benchmarks -v /etc/localtime:/etc/localtime:ro --privileged
horovod:latest-mlnxofed_gpudirect-tf1.14_cuda10.0 bash -c "/usr/sbin/sshd -p 50000;
sleep infinity"
```

On Primary node:

```
$ sudo docker run --gpus all -it --network=host -v /root/.ssh:/root/.ssh --cap-
add=IPC_LOCK -v /home/dell/imagenet_tfrecords/:/data/ -v
/home/dell/benchmarks/:/benchmarks -v /etc/localtime:/etc/localtime:ro --privileged
horovod:latest-mlnxofed_gpudirect-tf1.14_cuda10.0
```

- Running the benchmark in single node mode with 4 GPUs:

```
$ python /benchmarks/scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --
device=gpu --data_format=NCHW --optimizer=sgd --distortions=false --
use_fp16=True --local_parameter_device=gpu --variable_update=replicated --
all_reduce_spec=nccl --data_dir=/data/train --data_name=imagenet --
model=ResNet-50 --batch_size=256 --num_gpus=4 --xla=true
```

- Running the benchmark in multi node mode with 8 GPUs:

```
$ mpirun -np 8 -H 192.168.11.1:4,192.168.11.2:4 --allow-run-as-root -x
NCCL_NET_GDR_LEVEL=3 -x NCCL_DEBUG_SUBSYS=NET -x NCCL_IB_DISABLE=0 -mca
btl_tcp_if_include ib0 -x NCCL_SOCKET_IFNAME=ib0 -x NCCL_DEBUG=INFO -x
HOROVOD_MPI_THREADS_DISABLE=1 --bind-to none --map-by slot --mca plm_rsh_args
"-p 50000" python /benchmarks/scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --
device=gpu --data_format=NCHW --optimizer=sgd --distortions=false --
use_fp16=True --local_parameter_device=gpu  --variable_update=horovod --
horovod_device=gpu --datasets_num_private_threads=4 --data_dir=/data/train --
data_name=imagenet --display_every=10 --model=ResNet-50 --batch_size=256 --
xla=True
```