Deep Learning Performance Comparing Scale-out vs Scale-up

Abstract

This whitepaper looks at the performance and efficiency of Deep Learning training when using the Dell EMC PowerEdge C4140 server to run neural models. The objective is to show how the C4140 in scale-out configuration performs against scale-up server.

February 2019

Dell EMC Technical Whitepaper

Revisions

Date	Description
February 2019	Initial release

Acknowledgements

This paper was produced by the following persons:

Author:

Bhavesh Patel, Dell EMC Server Advanced Engineering.

Vilmara Sanchez, Dell EMC Server Advanced Engineering

Contributor: Josh Anderson, Dell EMC System Engineering

Contents

Acknowledgements

We would like to acknowledge the following individuals, Jaime Edwards (Director of PowerEdge Advanced Engineering) for setting the direction of this project, April Berman (PowerEdge Acceleration Product Manager), Shreya Shah (PowerEdge C4140 Product Manager) and Trevor Montgomery (Enterprise Systems Group Business Development) for providing us the resources for this paper.

We would also like to acknowledge:

- Google TensorFlow team for helping us in debugging issues related to TensorFlow.
- Uber Horovod team for explaining how to use their library in distributed training environment.
- Nvidia CUDA support team & Nvidia account team for their expedited support.
- Baidu Deep Bench support team
- Nimbix engineering and support team.

1 Overview

The objective of this whitepaper is to compare Dell’s PowerEdge acceleration optimized servers and determine their performance when running deep learning workloads. The purpose is to highlight how Dell’s scale out solution is ideally suited for these emerging workloads.

We will compare how PowerEdge C4140 performs using one of the more popular frameworks like TensorFlow with various neural architectures and compare it to other acceleration optimized servers in the market, targeting the same workloads. The idea is to investigate whether the architectural implementation helps PowerEdge C4140 in better utilizing the accelerators when running hardware level benchmarks like Baidu Deep bench and TensorFlow based benchmarks.

Using Baidu Deep bench, we can profile kernel operations, the lowest-level compute and communication primitives for Deep Learning (DL) applications. This allows us to profile how the accelerators are performing at the component level in different server systems. This is very important since it allows us to look at which hardware provides the best performance on the basic operations used for deep neural networks. Deep Bench includes operations and workloads that are important to both training and inference.

Using TensorFlow as the primary framework, we compare the performance in terms of throughput, and training time to achieve certain accuracy on ImageNet dataset. We look at performance at a single node level and multi-node level. We use some of the popular neural architectures like ResNet-50, VGG Net, GoogLeNet, and AlexNet to do this performance comparison.

1.1 Definition

Scale up : Scale up is achieved by putting the workload on a bigger, more powerful server (e.g., migrating from a two-socket server to a four- or eight-socket x86 server in a rack-based or blade form factor). This is a common way to scale databases and several other workloads. It has the advantage of allowing organizations to avoid making significant changes to the workload; IT managers can just install the workload on a bigger box and keep running it the way they always have.

Scale out: Scale out refers to expanding to multiple servers rather than a single bigger server. The use of availability and clustering software (ACS) and its server node management, which enables IT managers to move workloads from one server to another or to combine them into a single computing resource, represents a prime example of scale out. It adds flexibility in allowing IT organizations to add nodes as the number of users or workloads increase and this helps in better control of IT budgets.

2 Introduction

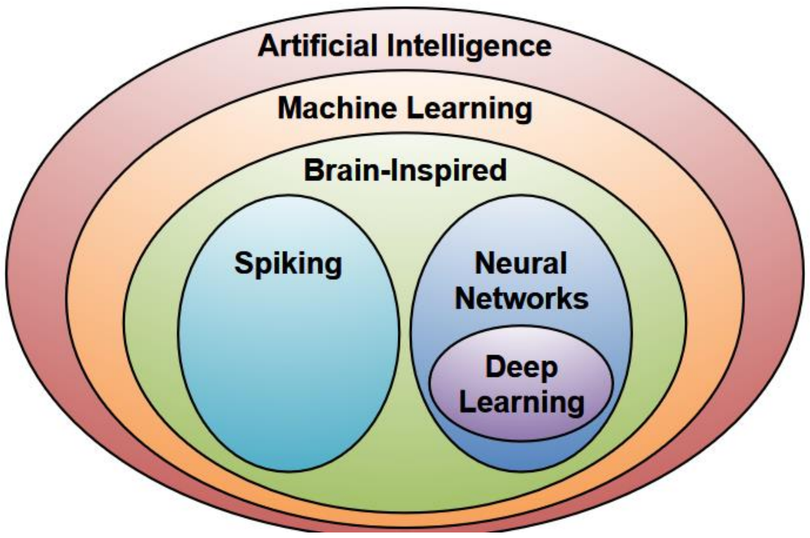


Figure 1: Artificial Intelligence, Machine Learning and Deep Learning [Source: MIT]

Artificial Intelligence

First coined in 1956 by John McCarthy, **AI involves machines that can perform tasks that are characteristic of human intelligence**. While this is rather general, it includes things like planning, understanding language, recognizing objects and sounds, learning, and problem solving.

Machine Learning

Arthur Samuel coined the phrase not too long after AI, in 1959, defining it as, “the ability to learn without being explicitly programmed.” You see, you can get AI **without** using machine learning, but this would require building millions of lines of codes with complex rules and decision-trees.

So instead of hard coding software routines with specific instructions to accomplish a task, machine learning is a way of “training” an algorithm so that it can **learn** how. “Training” involves feeding massive amounts of data to the algorithm and allowing the algorithm to adjust itself and improve.

Spiking Neural Networks

Spiking neural networks (**SNNs**) are artificial neural network models that more closely mimic natural neural networks. In addition to neuronal and synaptic state, SNNs also incorporate the concept of time into their operating model. The idea is that neurons in the SNN do not fire at each propagation cycle (as it happens with typical multi-layer perceptron networks), but rather fire only when a membrane potential – an intrinsic quality of the neuron related to its membrane electrical charge – reaches a specific value. When a neuron fires, it generates a signal which travels to other neurons which, in turn, increase or decrease their potentials in accordance with this signal.

2.1 Deep Learning

Deep Learning consists of two phases: Training and inference. As illustrated in *Figure 2*, training involves learning a neural network model from a given training dataset over a certain number of training iterations and loss function. The output of this phase, the learned model, is then used in the inference phase to speculate on new data [1].

The major difference between training and inference is training employs *forward propagation* and *backward propagation* (two classes of the deep learning process) whereas inference mostly consists of forward propagation. To generate models with good accuracy, the training phase involves several training iterations and substantial training data samples, thus requiring many-core CPUs or GPUs to accelerate performance.

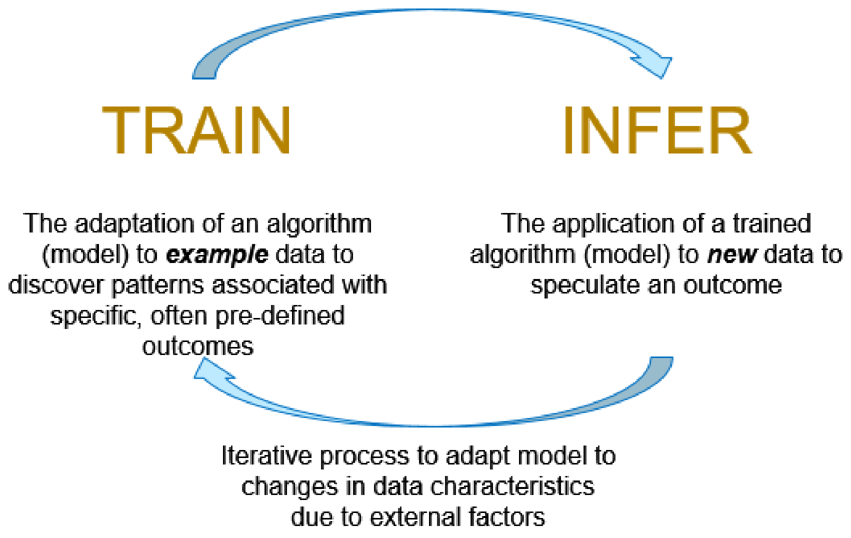

TRAIN
INFER
The adaptation of an algorithm (model) to example data to discover patterns associated with specific, often pre-defined outcomes
The application of a trained algorithm (model) to new data to speculate an outcome
Iterative process to adapt model to changes in data characteristics due to external factors

Figure 2. Deep Learning phases

3 Background

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 costly and often it comes down to what is the performance difference between a system with & without GPU.

The plot in *Figure 3* shows GPU performance when looking at single precision and *Figure 4* shows GPU performance when looking at half-precision. Most of the Deep Learning frameworks and models take advantage of half-precision since they can work with larger datasets with the available memory. It's very important to look at the raw Flop numbers for a GPU, since we want to extract the same level of performance when that GPU is put into a system.

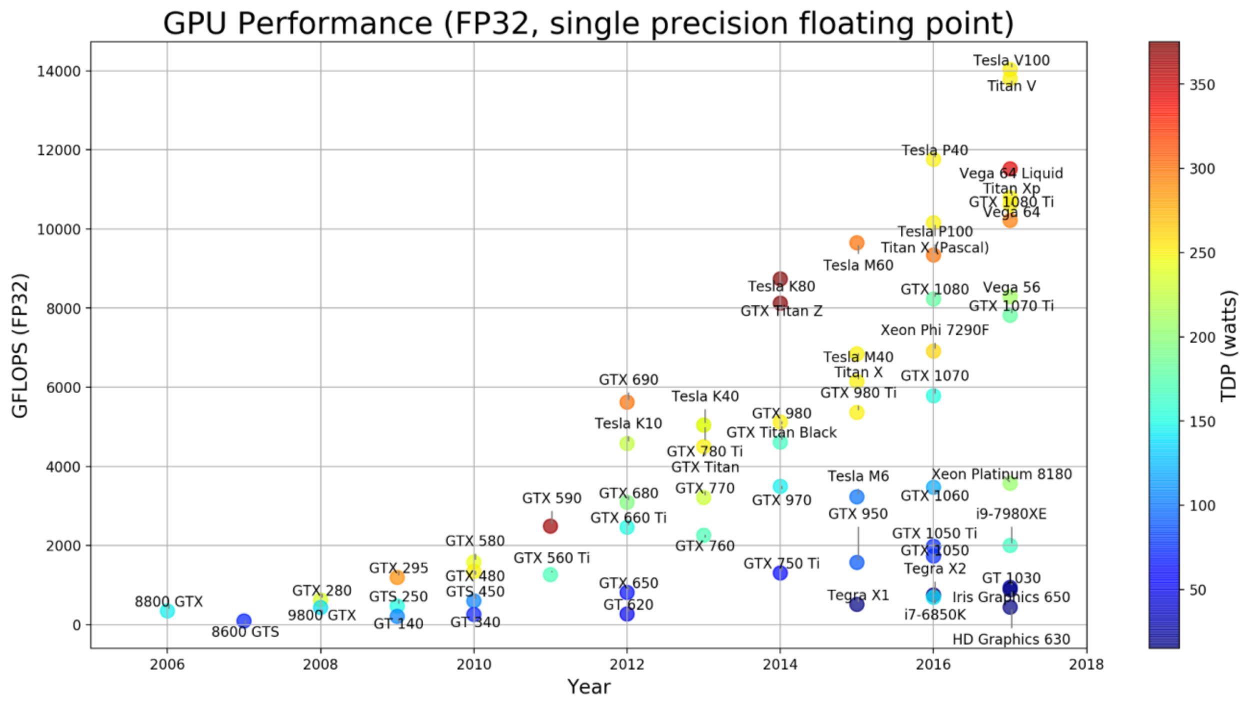


Figure 3 NVidia GPU Performance - Single precision [7]

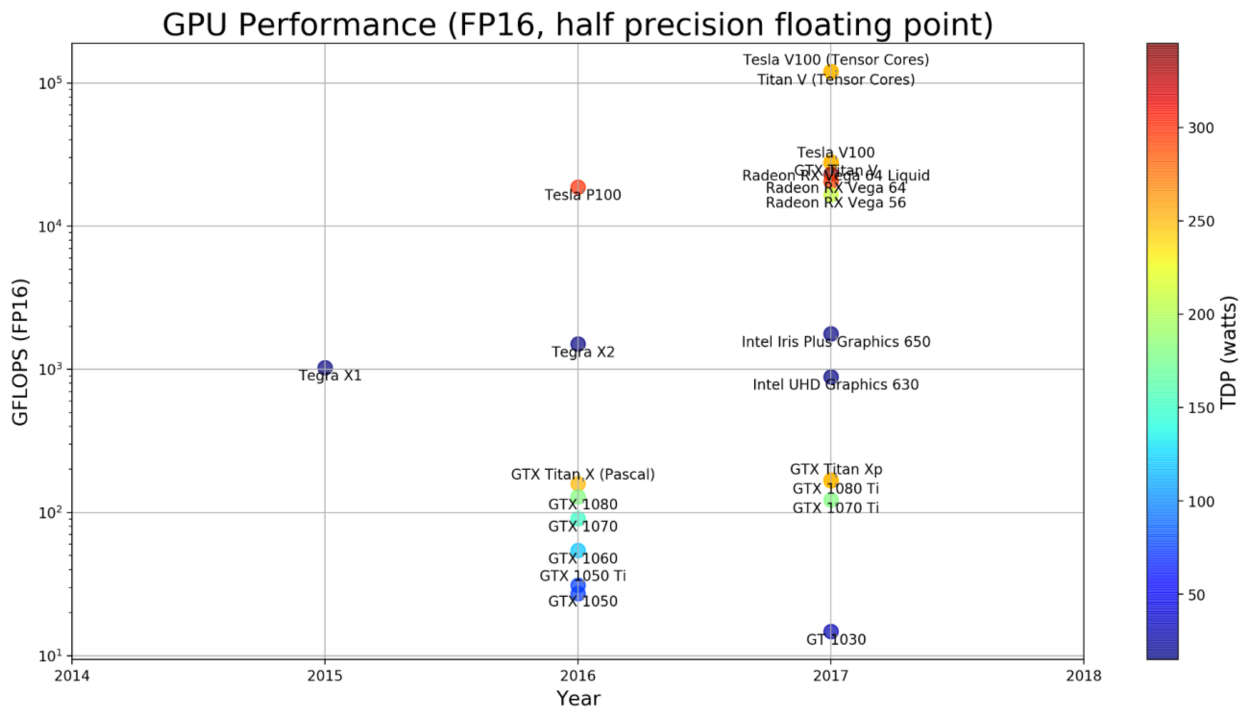
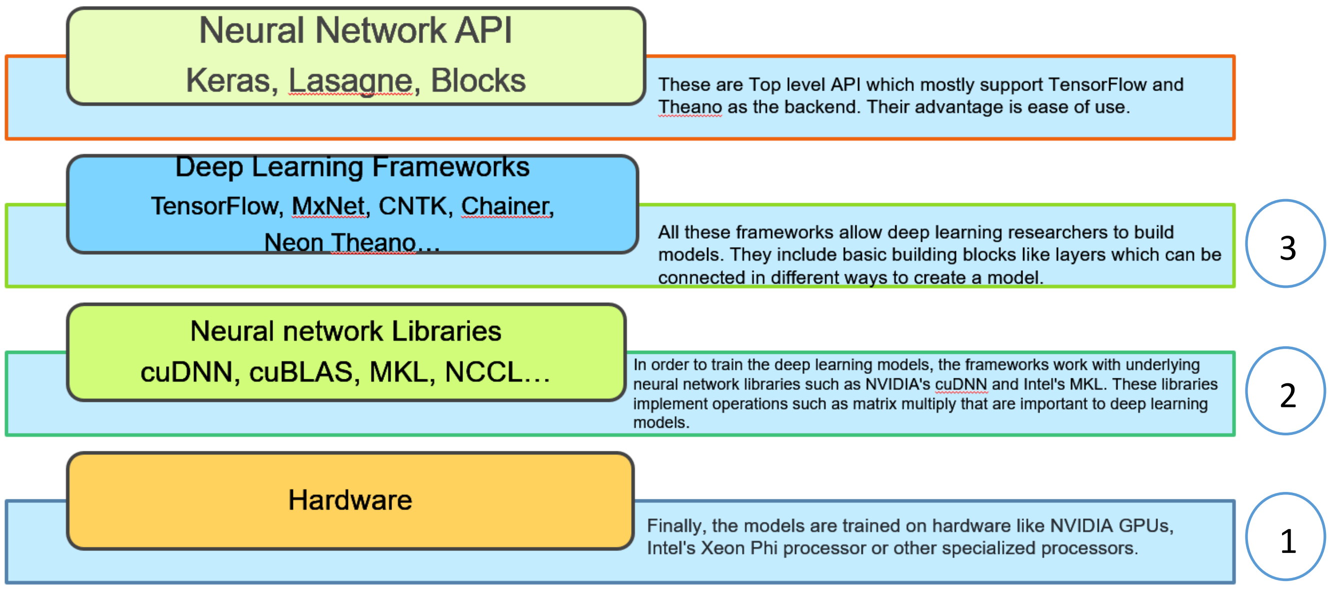


Figure 4 : GPU performance - Half-precision [7]

In order to see whether Dell EMC PowerEdge servers can meet the raw Flop numbers indicated in the charts above, we approached the problem by breaking it up into different sections. The picture below better illustrates how we are approaching this testing.



These are Top level API which mostly support TensorFlow and Theano as the backend. Their advantage is ease of use.

All these frameworks allow deep learning researchers to build models. They include basic building blocks like layers which can be connected in different ways to create a model.

In order to train the deep learning models, the frameworks work with underlying neural network libraries such as NVIDIA's cuDNN and Intel's MKL. These libraries implement operations such as matrix multiply that are important to deep learning models.

Finally, the models are trained on hardware like NVIDIA GPUs, Intel's Xeon Phi processor or other specialized processors.

Figure 5: Machine Learning Stack

1. System bandwidth performance i.e. PCIe connected to GPU - p2pbandwidth & latency tests
2. GPU hardware performance without any Deep learning frameworks – Baidu Deep Bench
3. System running GPU & benchmarks – TensorFlow benchmarks

3.1 Criteria

1. In order to bound our testing, we picked TensorFlow as the framework of choice since it has better support and models are readily available.
2. For distributed training, we selected Uber Horovod implementation, since it's one of the best performing distributed implementation [2].

3.2 Why TensorFlow as the framework of choice?

The reason we selected TensorFlow is because it's the most widely used framework of choice for machine learning and deep learning. It also has a wider support within open source community and availability of pre-trained models. It also has better community support and supported very well by the TensorFlow team.

TensorFlow is also widely used within the Dell EMC customer base and one of the top choices when developing any new projects in machine learning. Figure 6 shows how TensorFlow compares in terms of GitHub commits, stars and number of forks. This is a pretty good indicator of its widespread adoption.

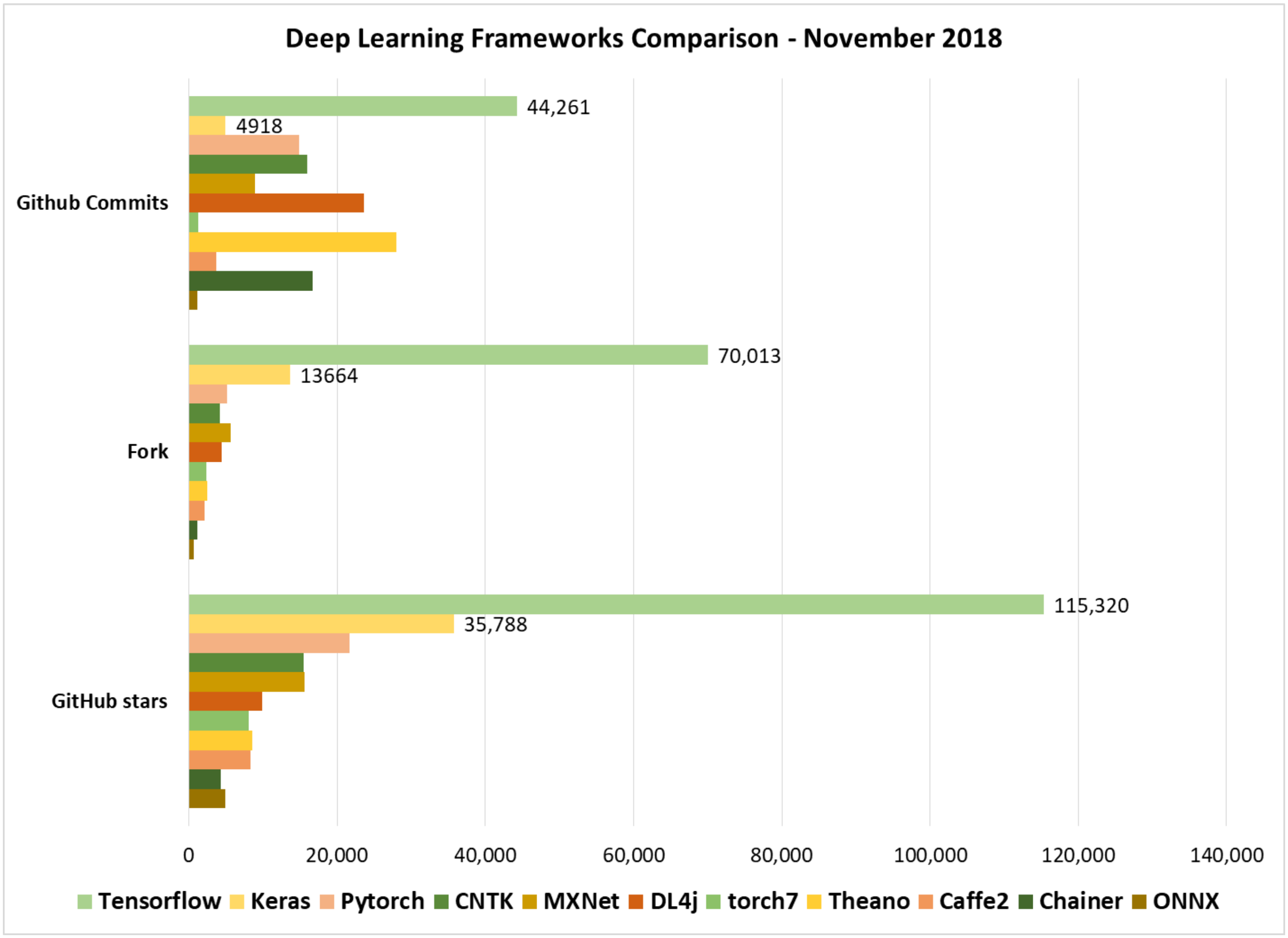


Figure 6: Frameworks Comparison

4 Test Methodology

The test methodology consists of 3 distinct phases.

Phase 1 is where we test the hardware performance of each server using NVidia supplied p2pbandwidth and latency tests and Baidu Deep Bench. This is explained in section Phase 1.

Phase 2 we used TensorFlow framework and ran some of the well-known neural models to compare performance in terms of throughput & training time. This is explained in section Phase 2.

Phase 3 we used TensorFlow but compared performance at multi-node level using Uber's Horovod implementation.

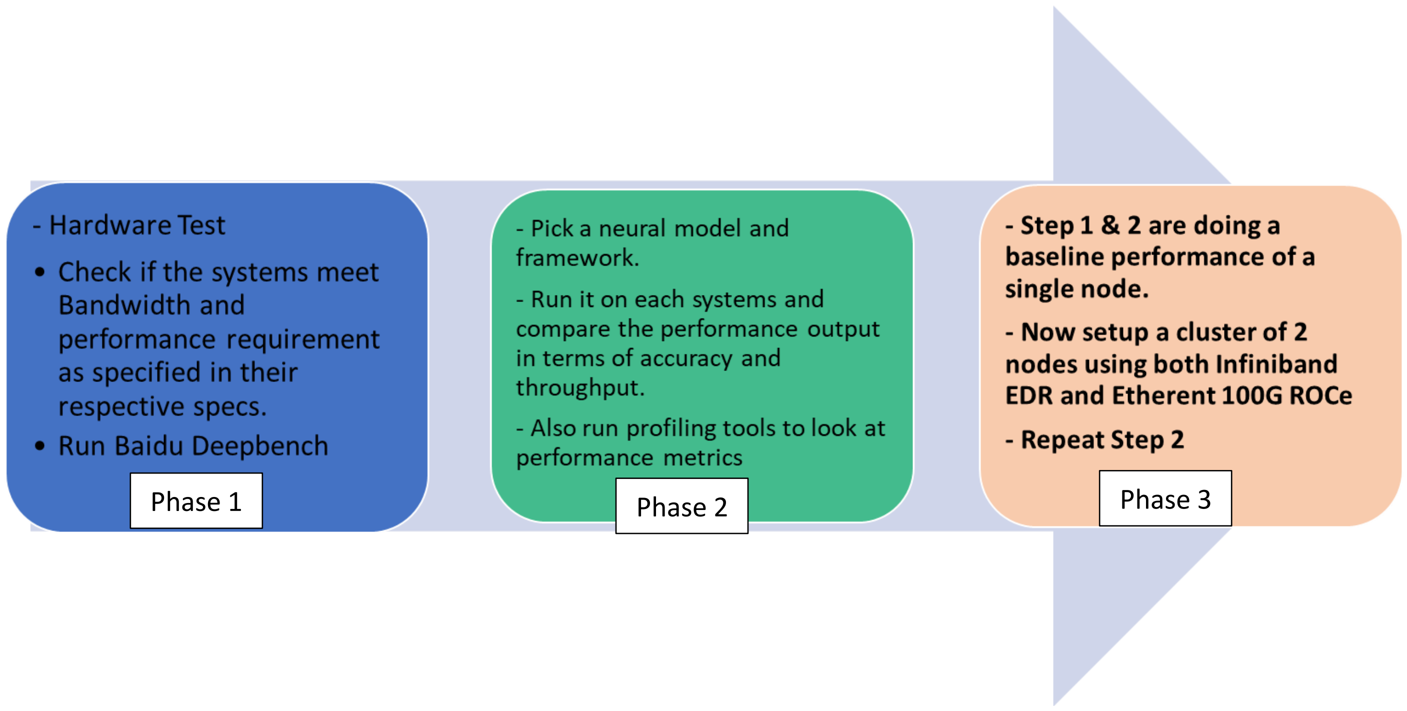

- Hardware Test
• Check if the systems meet Bandwidth and performance requirement as specified in their respective specs.
• Run Baidu Deepbench
Phase 1
- Pick a neural model and framework.
- Run it on each systems and compare the performance output in terms of accuracy and throughput.
- Also run profiling tools to look at performance metrics
Phase 2
- Step 1 & 2 are doing a baseline performance of a single node.
- Now setup a cluster of 2 nodes using both Infiniband EDR and Etherent 100G ROCe
- Repeat Step 2
Phase 3

Figure 7: Testing Methodology Workflow

1) **Phase 1** – In this phase we are performing some of the basic tests like PCIe bandwidth and latency tests to ensure it aligns to what we expect based on theoretical numbers. We then ran Baidu Deep Bench Benchmarks to evaluate deep learning performance for the accelerators and the system. The results for this step are presented in a separate whitepaper.
2) **Phase 2** - We used the official TensorFlow benchmarks which were run across several servers.
3) **Phase 3** – To benchmarks the servers in the distributed mode we used Horovod, which is a distributed training framework for TensorFlow.

4.1 Testing Methodology

We cover the testing methodology for Phase 1 in detail with its results in a separate whitepaper. Here we will describe the methodology we used in Phase 2 and Phase 3.

To establish a baseline, we divided it into short tests and long tests.

4.1.1 Short Test

The short tests consisted of 10 warmup steps and then another for another 100 steps which were averaged to get the actual throughput. The benchmarks were run with 1 GPU to establish a baseline number of images/sec and then increasing the number of GPUs under test based on the number of GPUs supported.

4.1.2 Long Test

The long tests were run to get throughput and the training time to reach certain accuracy convergence. We used 90 epochs for training run. These tests were run using the maximum number of GPUs supported by that server.

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

- **Use Case** – The benchmark tests are targeting image classification with convolutional neural networks models (CNNs).
- **Benchmark code** – TensorFlow Benchmarks scripts
- **Hardware Configuration** – Each server is configured based on its maximum GPU support.
- **Servers** - The servers tested are PowerEdge R740, PowerEdge C4130, PowerEdge C4140 and non-Dell EMC 8x NVLink GPU server.
- **Frameworks** – TensorFlow for single node, and TensorFlow with Horovod library for distributed training.
- **Performance** – The performance metrics used for comparison across servers is throughput (images per second) and training time to reach top-5 accuracy and top-1 accuracy.
- **Training tests** - We conducted two types of tests. 1- Short Tests: for each test, 10 warmup steps were done and then the next 100 steps were averaged. 2-Long Tests: to get the training accuracy convergence, and elapsed training time.
- **Dataset** – ILSVRC2012
- **Software stack configuration** – The benchmarks were run under docker container environment. See table 1 with details.

4.2 Throughput Testing

Workload application and model	Image classification with convolutional neural networks models (CNNs)	
Benchmarks code	TensorFlow Benchmarks scripts	
	Server	GPU
Servers – Single Node	▪ PowerEdge R740	▪ P40
	▪ PowerEdge C4140	▪ V100-16GB-SXM2
	▪ PowerEdge C4140	▪ V100-32GB-SXM2
	▪ Non Dell EMC 8x NVLink server	▪ V100-16GB-SXM2
Servers – Multi Node (2 nodes, 4GPUs each)	▪ PowerEdge C4140-K	▪ V100-16GB-SXM2
	▪ PowerEdge C4140-K	▪ V100-32GB-SXM2
	▪ PowerEdge C4140-M	▪ V100-16GB-SXM2
Frameworks	▪ TensorFlow for Single Mode ▪ TensorFlow with Horovod library for Distributed Mode	

Performance Metrics	▪ Throughput images/second ▪ Top-5 Accuracy on the training dataset ▪ Training time
Training Tests	▪ Short Tests to get throughput images/second ▪ Long Tests to get accuracy convergence and training time
Dataset	ILSVRC2012

Table 1: Benchmark Setup

4.3 Neural Models & parameters

	Short Tests	**Long Tests**
Models	▪ VGG-16 ▪ VGG-19 ▪ Inception-v3 ▪ Inception-v4 ▪ ResNet50 ▪ GoogLeNet ▪ AlexNet	▪ VGG-19 ▪ Inception-v4 ▪ ResNet50
Batch sizes	▪ 32 ▪ 64 ▪ 128 ▪ 256 ▪ 512 ▪ 2014 ▪ 2048	▪ 32 ▪ 64 ▪ 128 ▪ 256
GPUs	▪ 1 ▪ 2 ▪ 3 ▪ 4 ▪ 5 ▪ 6 ▪ 7 ▪ 8	▪ 3 ▪ 4 ▪ 8
Training Steps or Epochs	100 steps	90 epochs

Table 2: Neural Models & Parameters

5 PowerEdge Server Details

5.1 PowerEdge C4140

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 Volta SMX2 GPUs, both the V100-SXM2 as well as the V100-PCIe models.

Dell EMC PowerEdge C4140 supporting NVIDIA Volta SXM2 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 or 8-way or 16-way NVIDIA Volta SXM use PCIe bridges and this limits the total available bandwidth between CPU to GPU.

5.1.1 Why is C4140 Configuration-M better?

Configuration	Link Interface b/n CPU- GPU complex	Total Bandwidth	Notes
K	X16 Gen3	32GB/s	Since there is a PCIe switch between host to GPU complex
G	X16 Gen3	32GB/s	Since there is a PCIe switch between host to GPU complex
M	4x16 Gen3	128GB/s	Each GPU has individual x16 Gen3 to Host CPU

Table 3: Host-GPU Complex PCIe Bandwidth comparison

As shown in *Table 3* the total available bandwidth between CPU – GPU complex is much higher than other configurations. This greatly benefits neural models in taking advantage of larger capacity although lower bandwidth DDR memory to speed up learning.

Figure 8 shows the CPU-GPU and GPU-GPU connection topology for C4140-K, *Figure 9* shows topology for C4140-M and *Figure 10* shows topology for C4140-B.

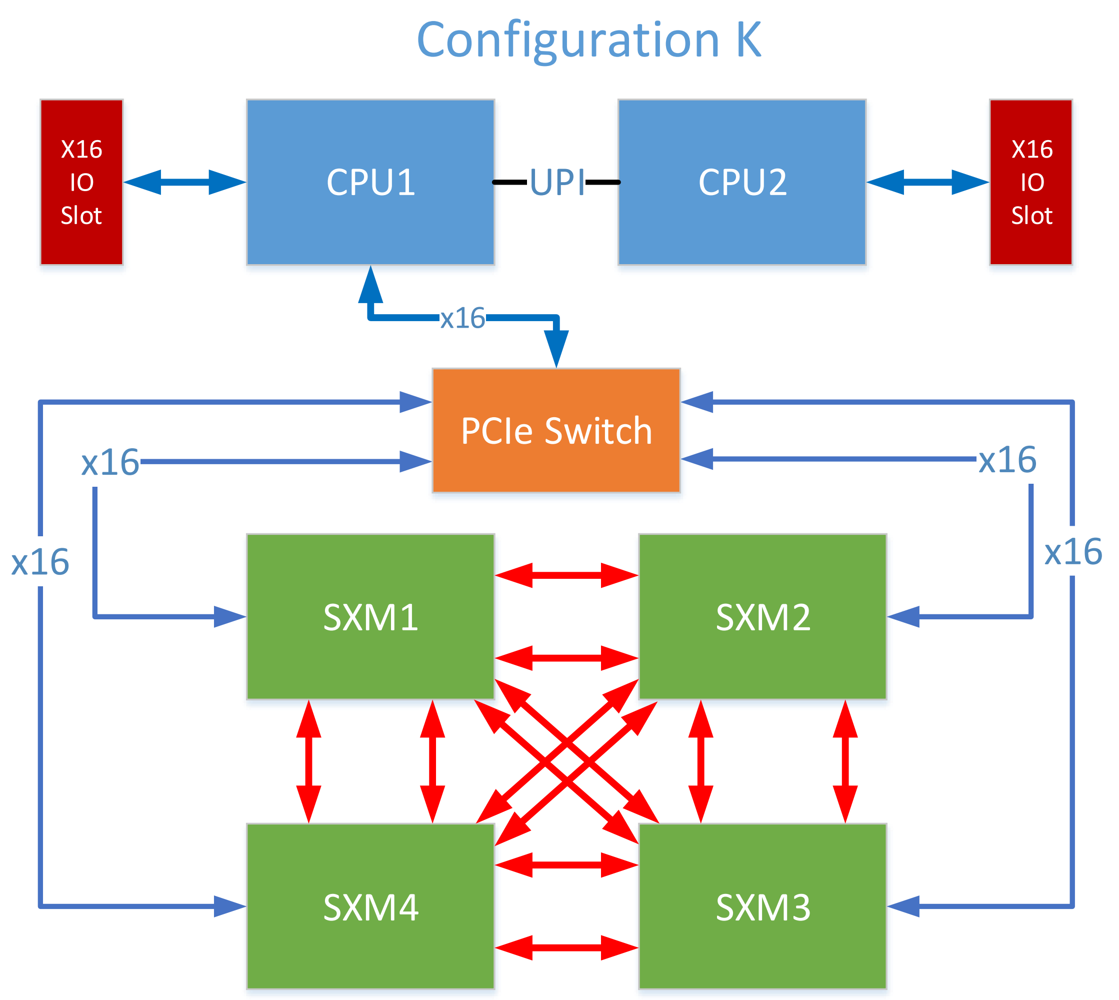
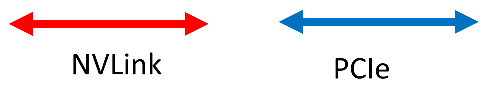


Figure 8: C4140 Configuration-K



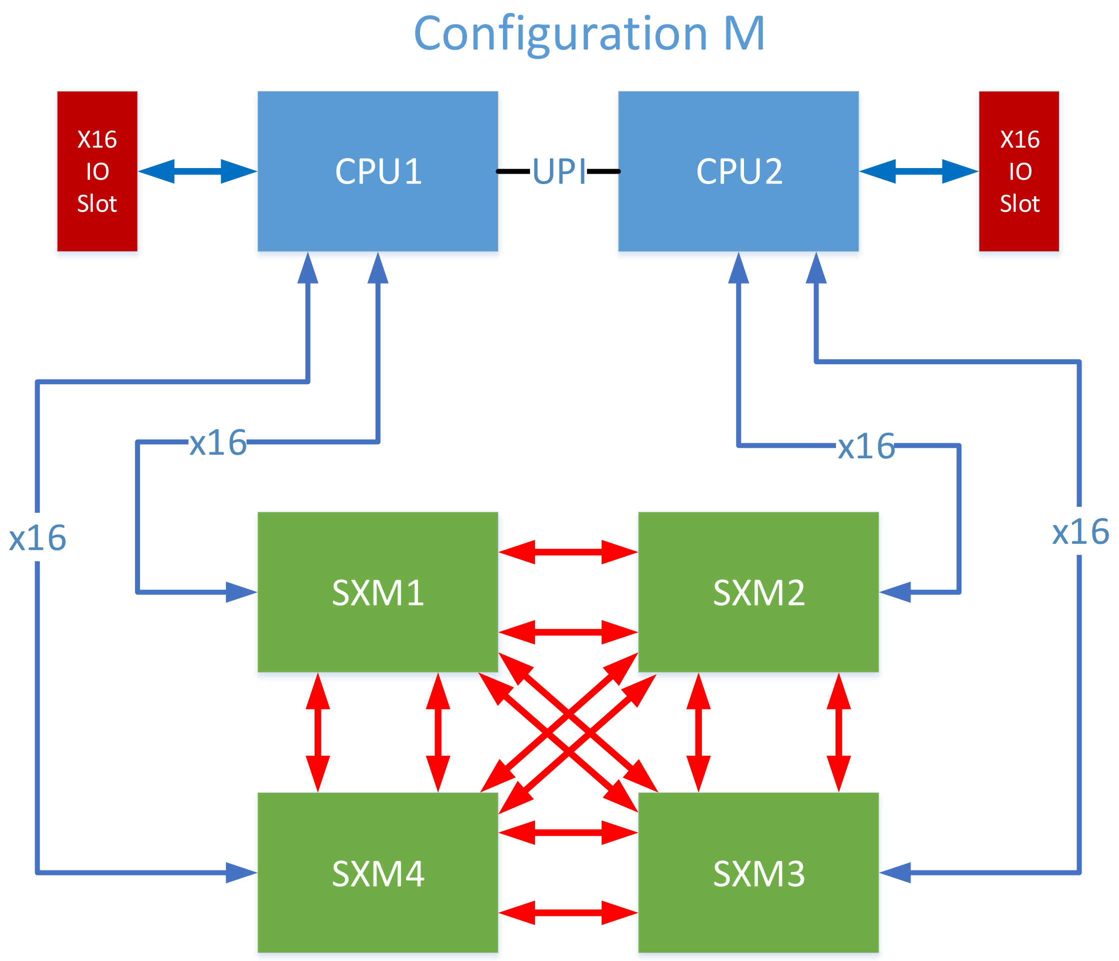
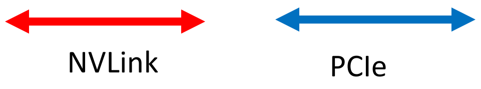


Figure 9: C4140 Configuration-M



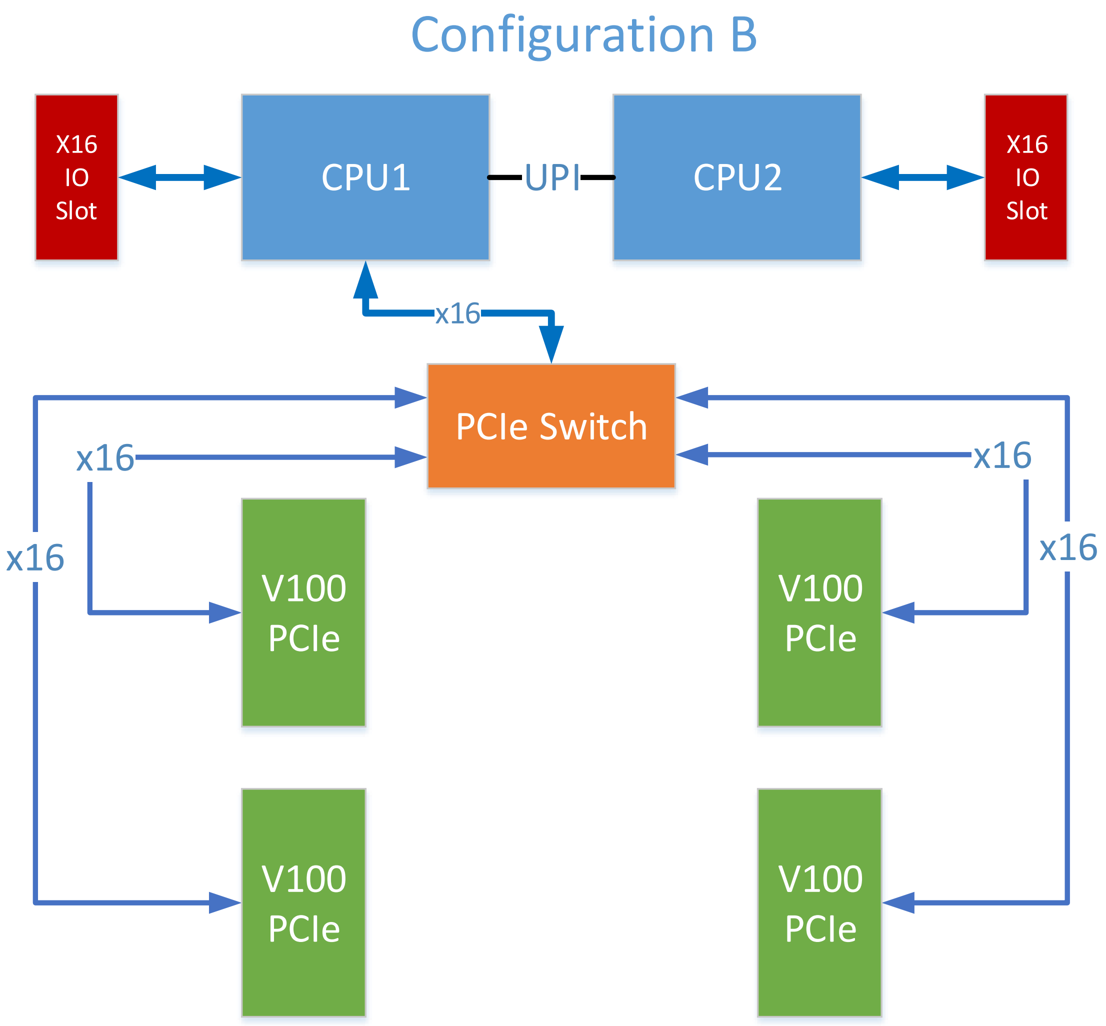


Figure 10 : PowerEdge C4140 Configuration B

↔

PCIe

5.2 PowerEdge R740/R740xd

The PowerEdge R740/R740xd is a general-purpose platform with highly expandable memory (up to 3TB) and impressive I/O capability to match both read-intensive and write-intensive operations. The R740 is capable of handling demanding workloads and applications such as data warehouses, E-commerce, databases, high-performance computing (HPC), and Deep learning workloads.

Figure 11 below shows the topology connection for CPU: GPU in PowerEdge R740/R740xd server.

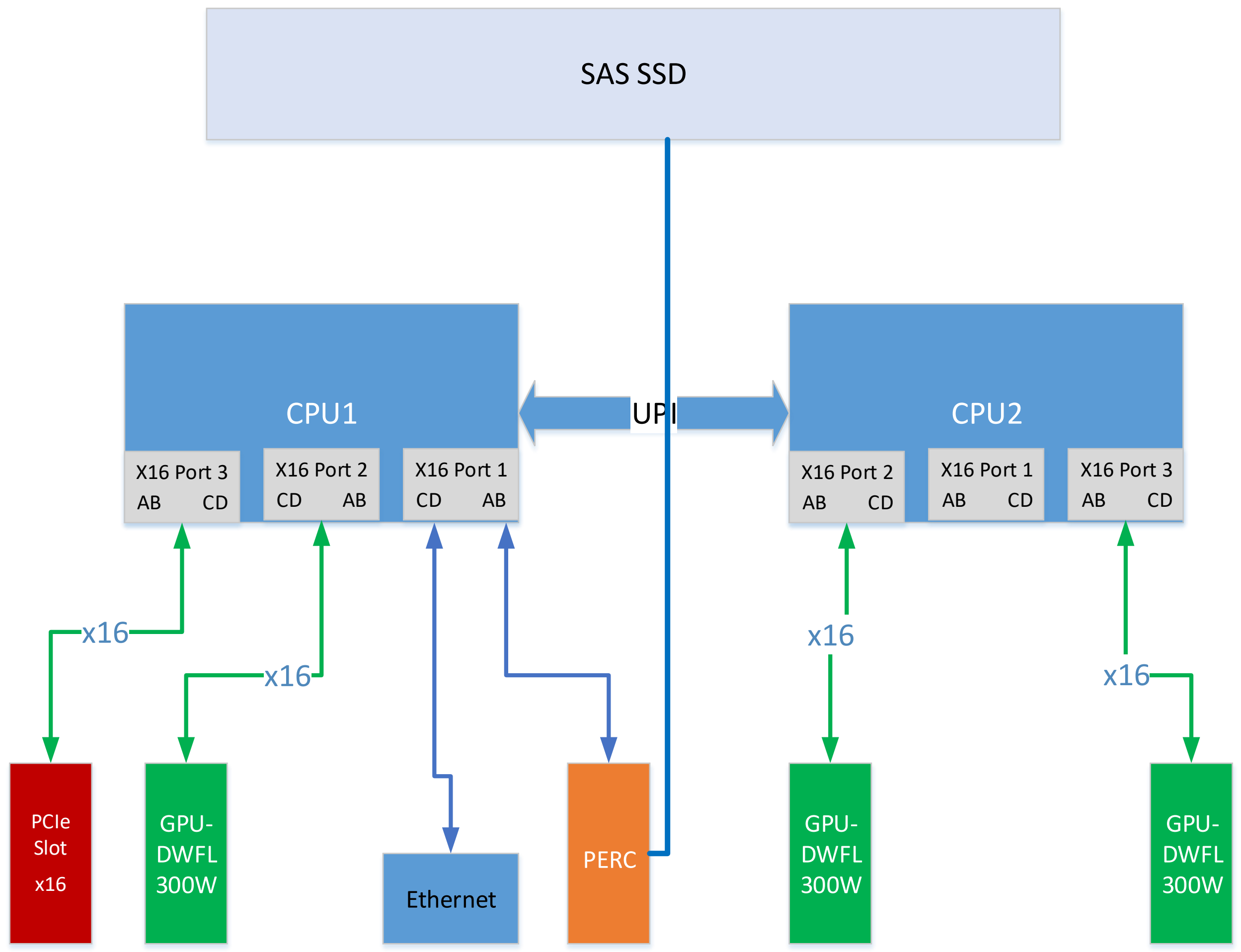


Figure 11: Dell PowerEdge R740/R740xd

6 Framework Setup Details

6.1 Distributed Horovod-TensorFlow Setup

Horovod [8] [9] [10] is a distributed training framework for TensorFlow, Keras and PyTorch initially developed by Uber. It uses bandwidth-optimal communication protocols (RDMA) [2]

In this section, we explain briefly the software stack configuration we used to extract the performance throughput in multi-node using distributed Horovod TensorFlow and using high speed Mellanox InfiniBand ConnectX-5 network adapter with 100Gbit/s over IPoIB, and GPUDirect RDMA.

To setup the configuration, we used as our reference the configuration procedure presented by Mellanox on its community blog space [3] and the basic installation of Horovod in Docker [4].

The tests were run in docker environment, *Figure 12* 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.

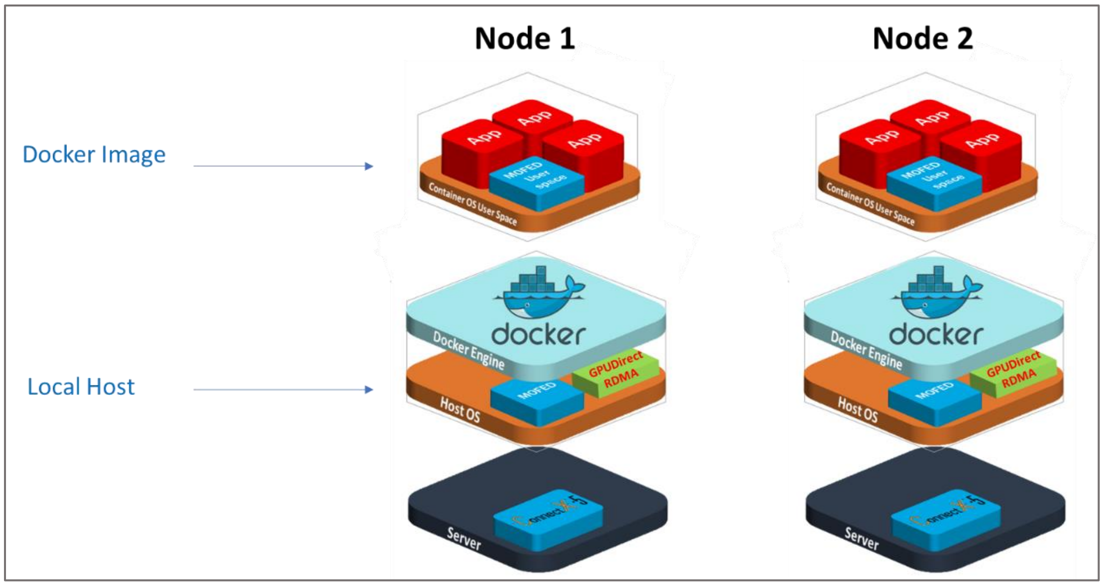


Figure 12: Servers Logical Design. Source: Image adapted from https://community.mellanox.com/docs/DOC-2971

In *Figure 13* below shows how PowerEdge C4130/C4140 is conncted via InifniBand fabric for multi-node testing.

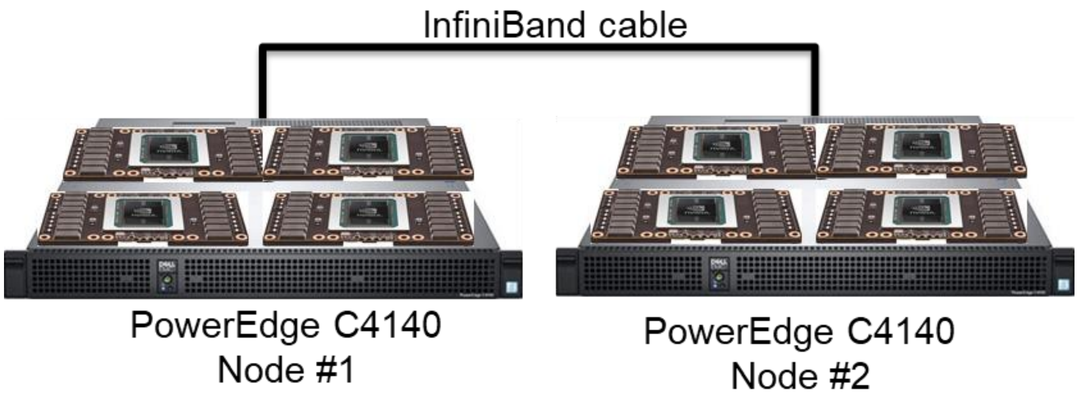


Figure 13: Using Mellanox CX5 InfiniBand adapter to connect C4130/PowerEdge C4140 in multi-node configuration

In *Figure 14* we see how the GPU memory is accessed directly instead of copying the data n times across the system components with the use of GPUDirect RDMA, this feature is reflected directly in the throughput performance of the server.

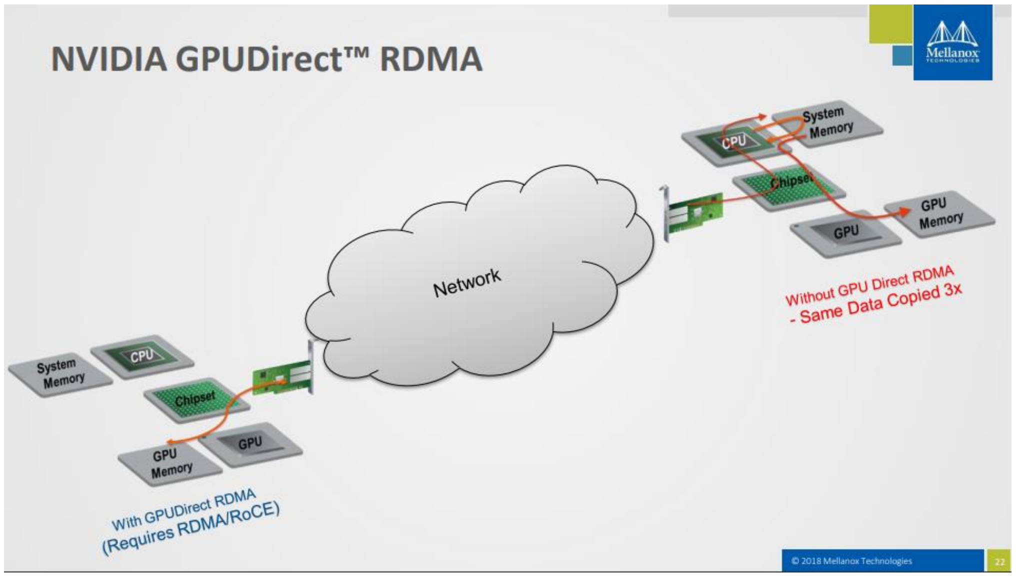

NVIDIA GPUDirect™ RDMA
Mellanox
System Memory
CPU
Chipset
GPU
GPU Memory
Network
Without GPU Direct RDMA - Same Data Copied 3x
With GPUDirect RDMA (Requires RDMA/RoCE)
© 2018 Mellanox Technologies
22

Figure 14: Nvidia GPU Direct RDMA Connection. Source: https://www.sc-asia.org

6.2 Evaluation Platform Setup

Table 4 shows the software stack configuration used to build the environment to run the tests.

Software Stack	PowerEdge Servers	Non-Dell EMC Servers
OS	Ubuntu 16.04.4 LTS	Ubuntu 16.04.3 LTS
Kernel	GNU/Linux 4.4.0-128-generic x86_64	GNU/Linux 4.4.0-130-generic x86_64
nvidia driver	396.26 for all servers 390.46 for R740-P40	384.145
Open MPI	3.0.1	3.0.0
CUDA	9.1.85	9.0.176
cuDNN	7.1.3.16	7.1.4
NCCL	2.2.15	2.2.13
Docker Container	NVidia TensorFlow Docker	Nvidia TensorFlow Docker
Container Image – Single Node	TensorFlow/tensorflow:nightly-gpu-py3	nvcr.io/nvidia/tensorflow:18.06-py3
Container Image – Multi Node	Horovod : latest	n/a
Benchmark scripts	tf_cnn_benchmarks	tf_cnn_benchmarks
Test Date – V1	April-June 2018	July 2018
Test Date - V2	Jan 2019	NA

Table 4: OS & Driver Configurations

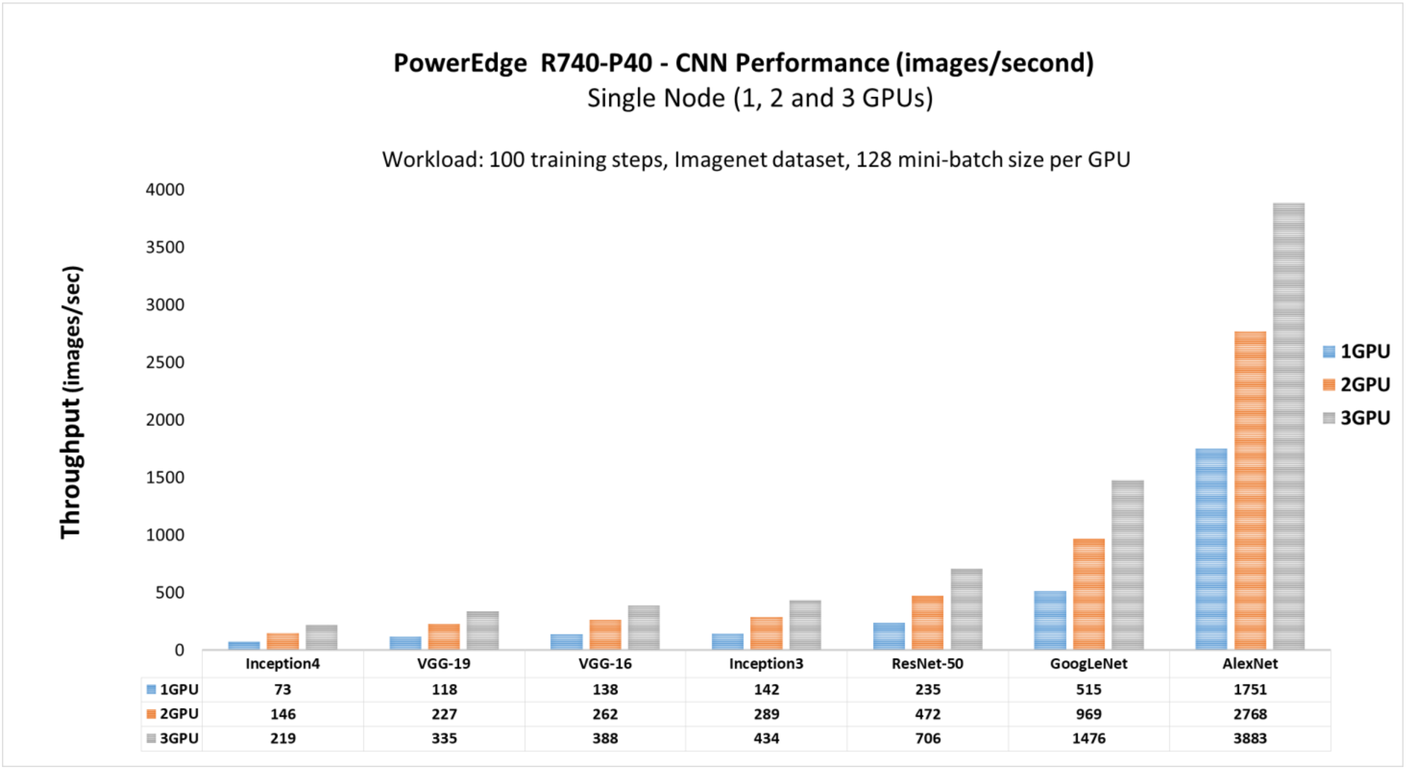
7 Performance Results

7.1 Single Node – Throughput (images/sec)

The charts below show the results for different servers running the short tests to extract throughput images/second using ResNet50 with batch size 128 and number of steps =100.

The results for single node are with maximum number of GPUs supported within that node.

7.1.1 PowerEdge R740xd

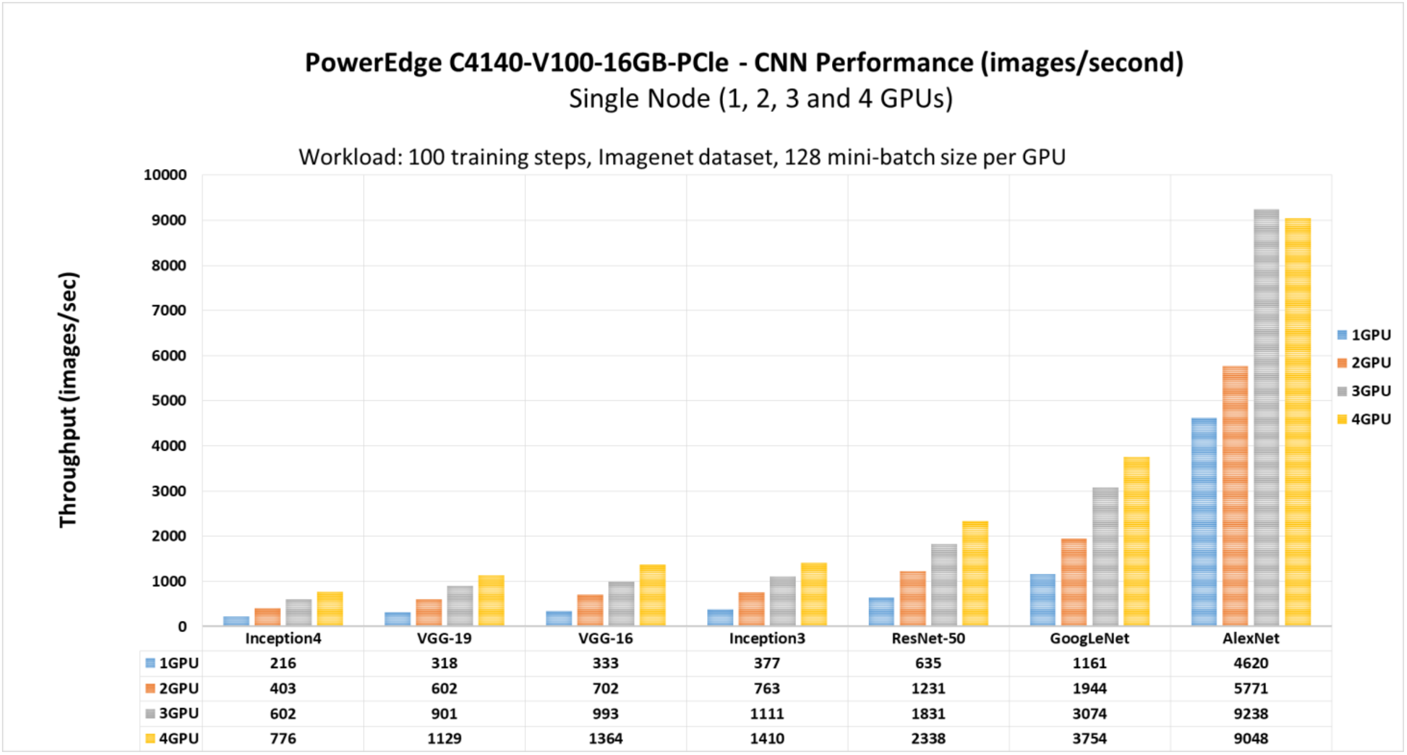


	Inception4	VGG-19	VGG-16	Inception3	ResNet-50	GoogLeNet	AlexNet
■ 1GPU	73	118	138	142	235	515	1751
■ 2GPU	146	227	262	289	472	969	2768
■ 3GPU	219	335	388	434	706	1476	3883

Figure 15: PowerEdge R740-P40 server with up to 3 GPUs

The PowerEdge R740 with P40 GPU is tested with different pre-trained neural models. The results show how different models use the amount of available memory e.g. ResNet50 uses more memory than GoogLeNet or AlexNet and therefore we see lower throughput.

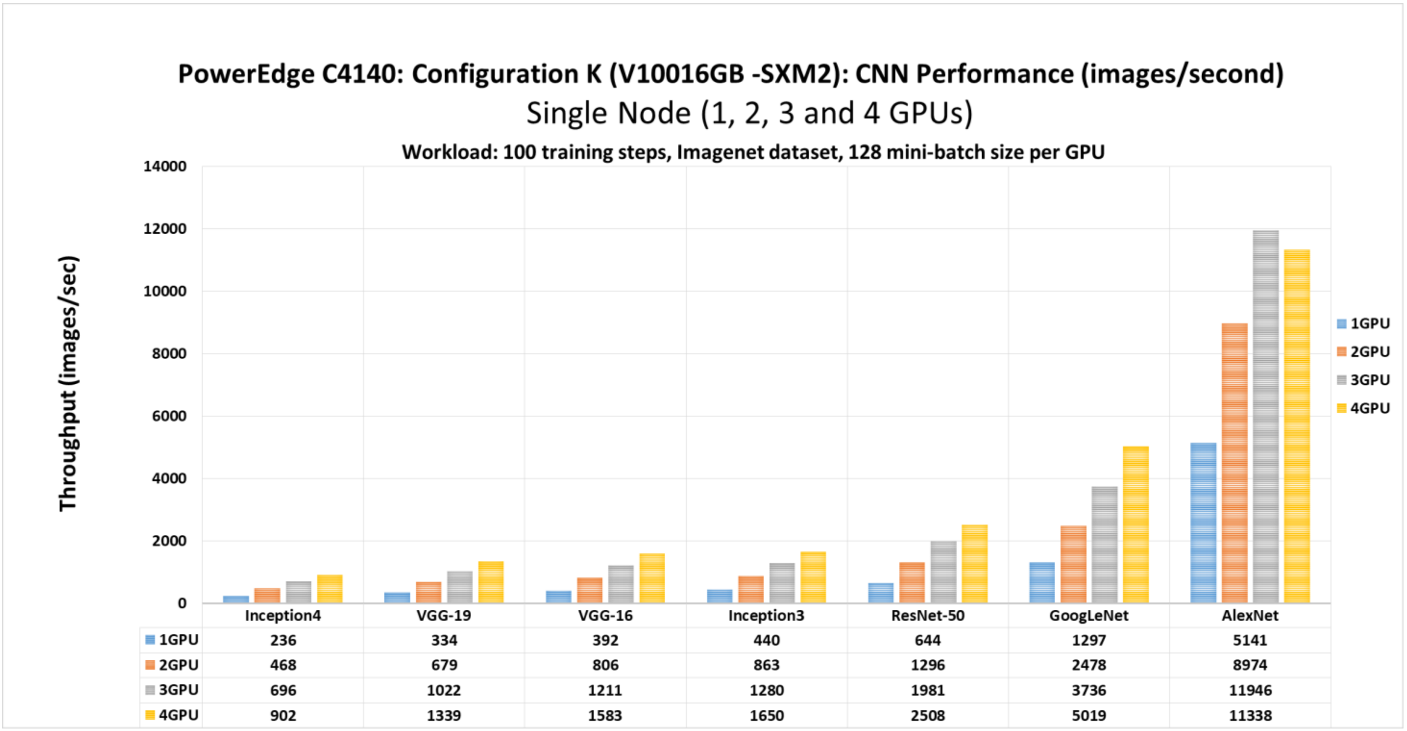
7.1.2 PowerEdge C4140-V100-16GB-PCle [Config B] – Single Node



	Inception4	VGG-19	VGG-16	Inception3	ResNet-50	GoogLeNet	AlexNet
1GPU	216	318	333	377	635	1161	4620
2GPU	403	602	702	763	1231	1944	5771
3GPU	602	901	993	1111	1831	3074	9238
4GPU	776	1129	1364	1410	2338	3754	9048

Figure 16: PowerEdge C4130-P100-16GB-PCle in single-node

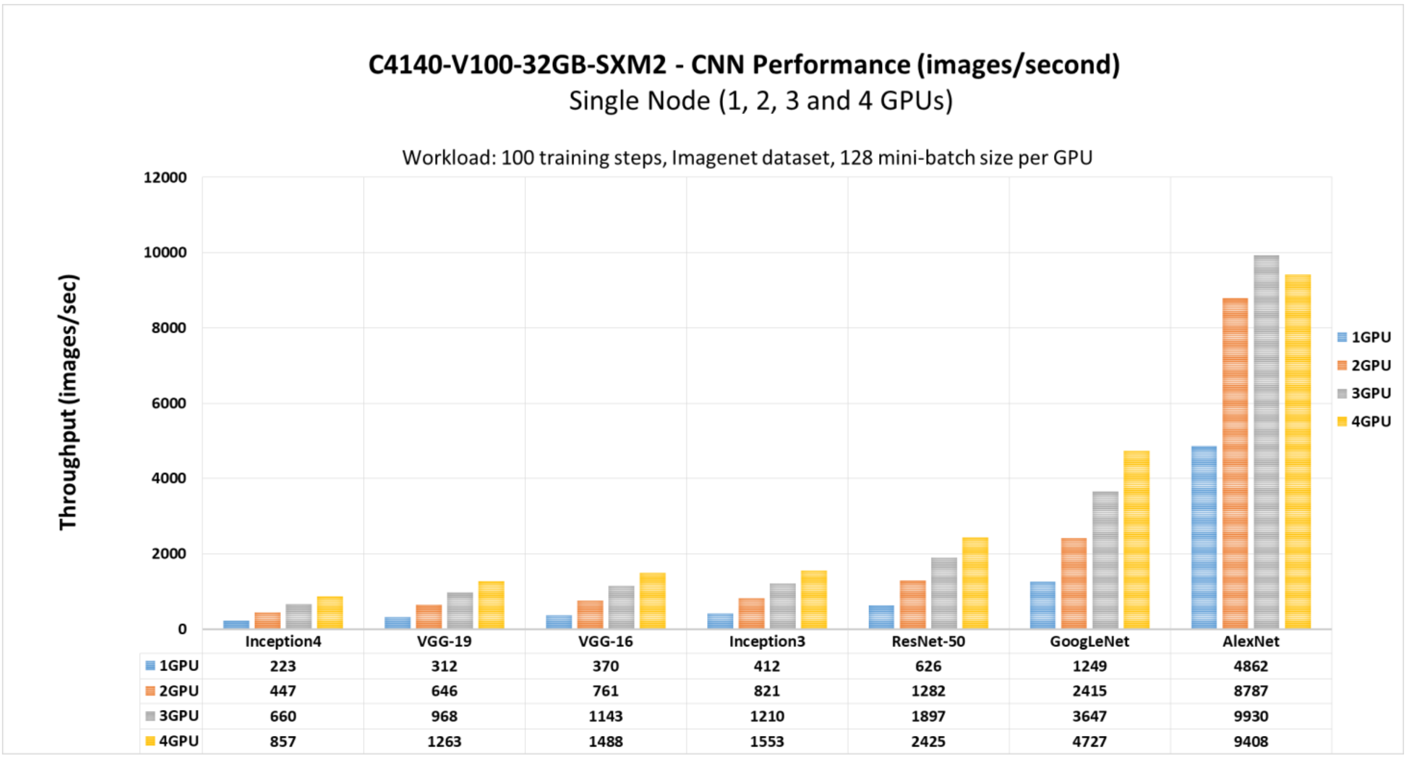
7.1.3 PowerEdge C4140-K-V100-16GB SXM2 Single Node



	Inception4	VGG-19	VGG-16	Inception3	ResNet-50	GoogLeNet	AlexNet
1GPU	236	334	392	440	644	1297	5141
2GPU	468	679	806	863	1296	2478	8974
3GPU	696	1022	1211	1280	1981	3736	11946
4GPU	902	1339	1583	1650	2508	5019	11338

Figure 17: PowerEdge C4140-V100-16GB-SXM2 in single-node

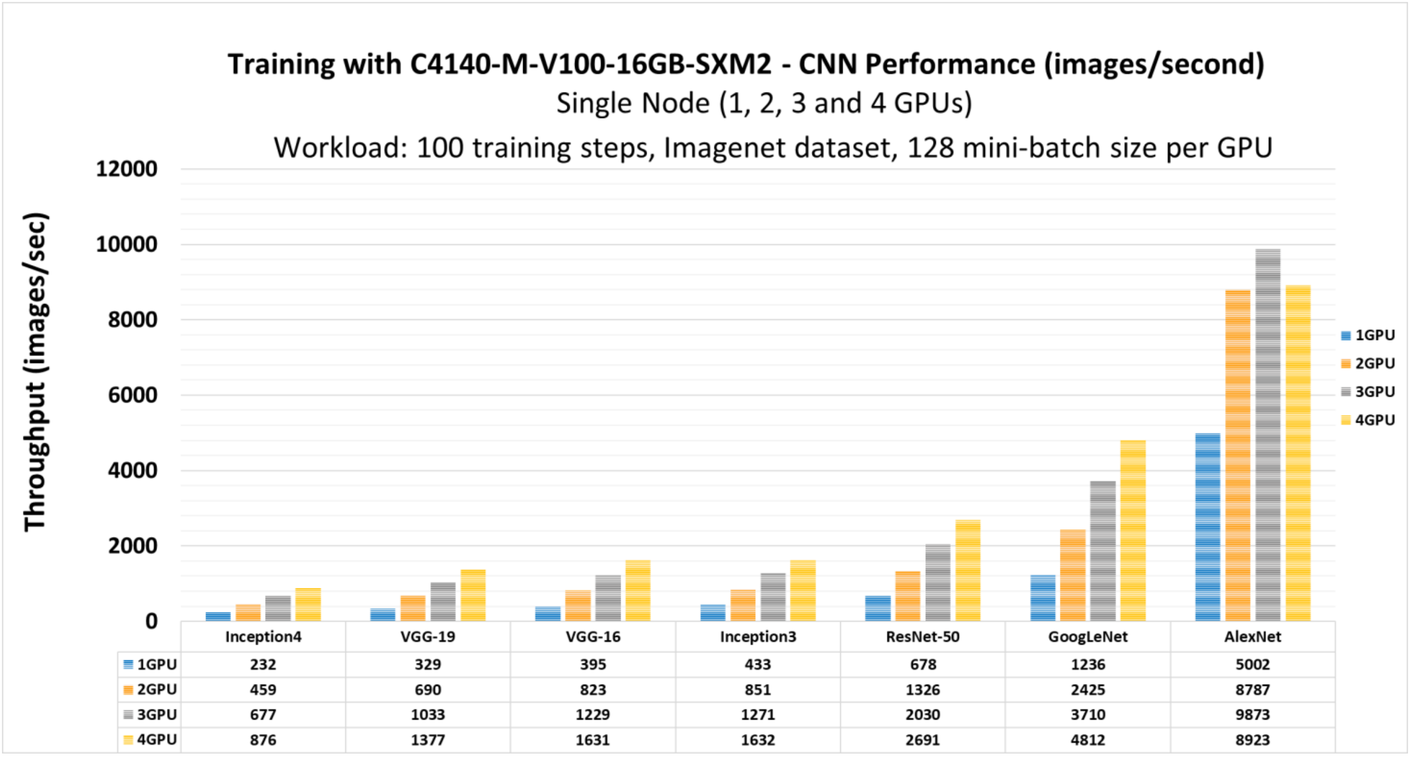
7.1.4 PowerEdge C4140-K- V100-32GB SXM2 Single Node



	Inception4	VGG-19	VGG-16	Inception3	ResNet-50	GoogLeNet	AlexNet
1GPU	223	312	370	412	626	1249	4862
2GPU	447	646	761	821	1282	2415	8787
3GPU	660	968	1143	1210	1897	3647	9930
4GPU	857	1263	1488	1553	2425	4727	9408

Figure 18: PowerEdge C4140-V100-32GB-SXM2 in single-node

7.1.5 PowerEdge C4140-M- V100-16GB SXM2 Single Node



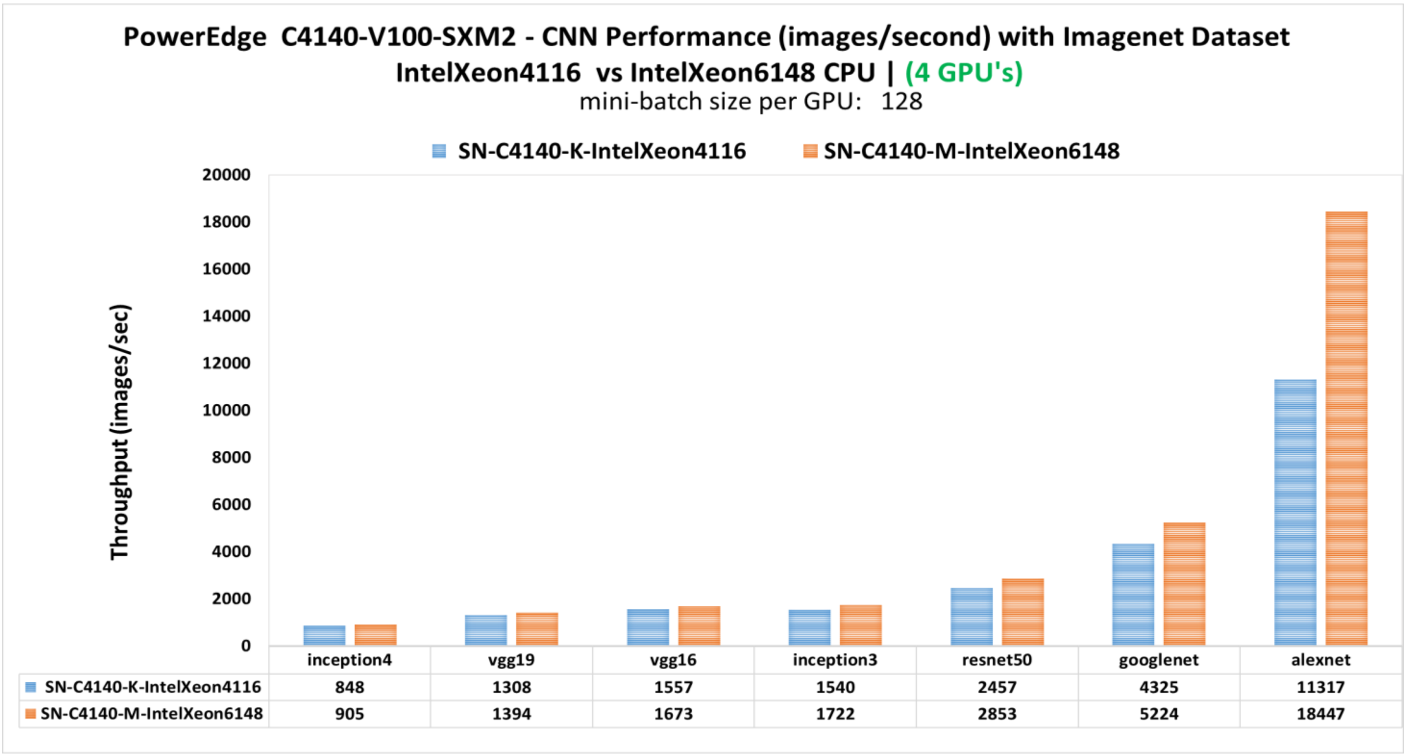
	Inception4	VGG-19	VGG-16	Inception3	ResNet-50	GoogLeNet	AlexNet
1GPU	232	329	395	433	678	1236	5002
2GPU	459	690	823	851	1326	2425	8787
3GPU	677	1033	1229	1271	2030	3710	9873
4GPU	876	1377	1631	1632	2691	4812	8923

Figure 19. PowerEdge C4140-M-V100-16GB-SXM2 in single-node

7.1.6

7.1.7 PowerEdge C4140- V100-SXM2 Configuration K versus Configuration M - Single Node

The plot below is a comparison showing performance difference between Config-K and Config-M, although it's not a like-like comparison because of CPU difference. Config-K has Intel Xeon 4116 @ 2.1GHz 12core processor Vs Config-M which has Intel Xeon 6148 @2.4GHz 20 cores. The number of CPU cores does matter and in one of the later plots, we will show the difference in performance using different processors with same PCIe topology.



	inception4	vgg19	vgg16	inception3	resnet50	googlenet	alexnet
🟦 SN-C4140-K-IntelXeon4116	848	1308	1557	1540	2457	4325	11317
🟧 SN-C4140-M-IntelXeon6148	905	1394	1673	1722	2853	5224	18447

Figure 20: PowerEdge C4140-V100-SXM2- Configuration-K vs PowerEdge C4140-V100-SXM2 Configuration-M

As shown in *Figure 21* below, it shows that the number of CPU cores does play a role in terms of throughput. And the biggest difference is when running AlexNet.

7.1.8 What role does CPU play in Deep learning?

The CPU plays a major role in the initial phase called data preprocessing. The steps below show an instruction pipeline, with the following 4 instructions happening in parallel:

a. Train on batch n (on GPUs)
b. Copy batch n+1 to GPU memory
c. Transform batch n+2 (on CPU)
d. Load batch n+3 from disk (on CPU)

The loop for the data processing when training is:

a. Load mini-batch
b. Preprocess mini-batch
c. Train on mini-batch

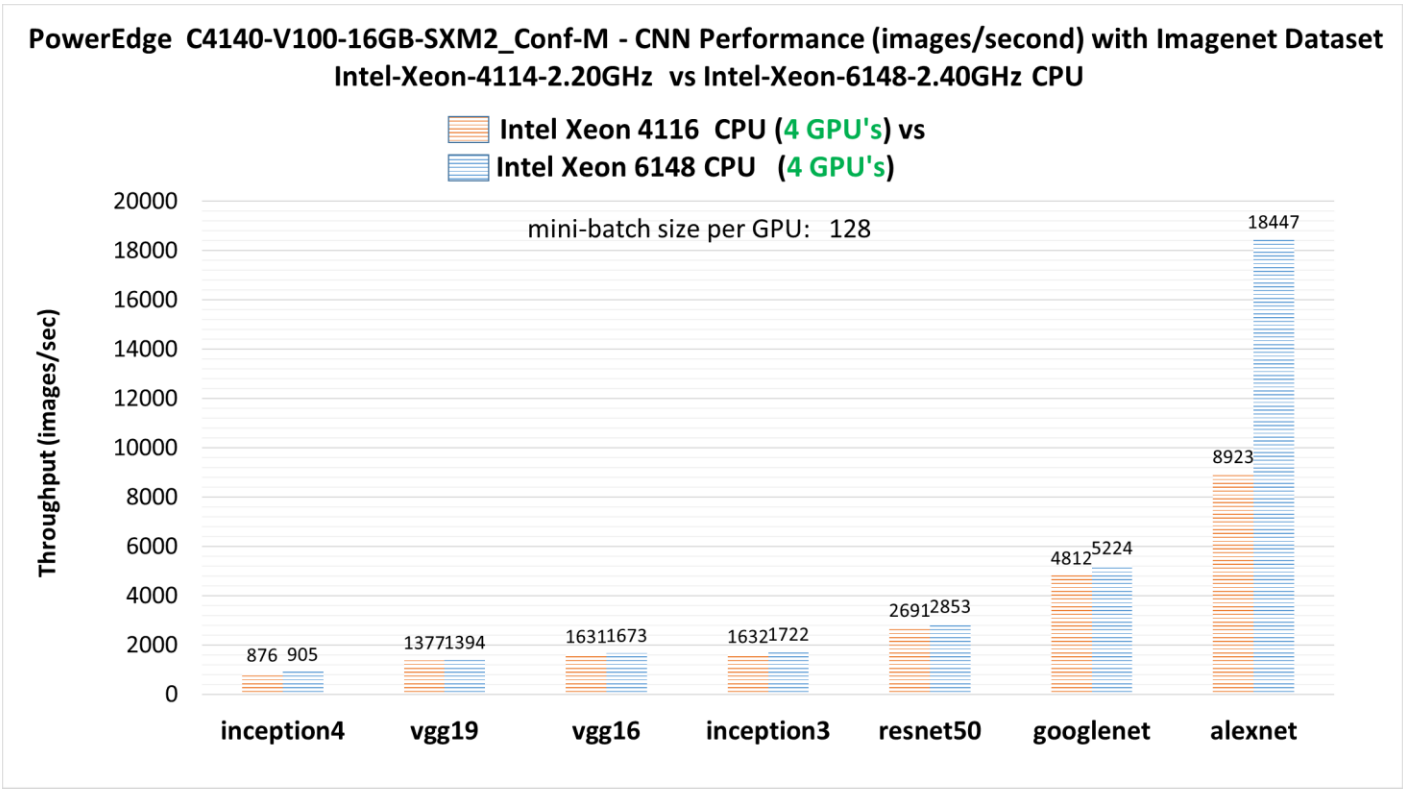
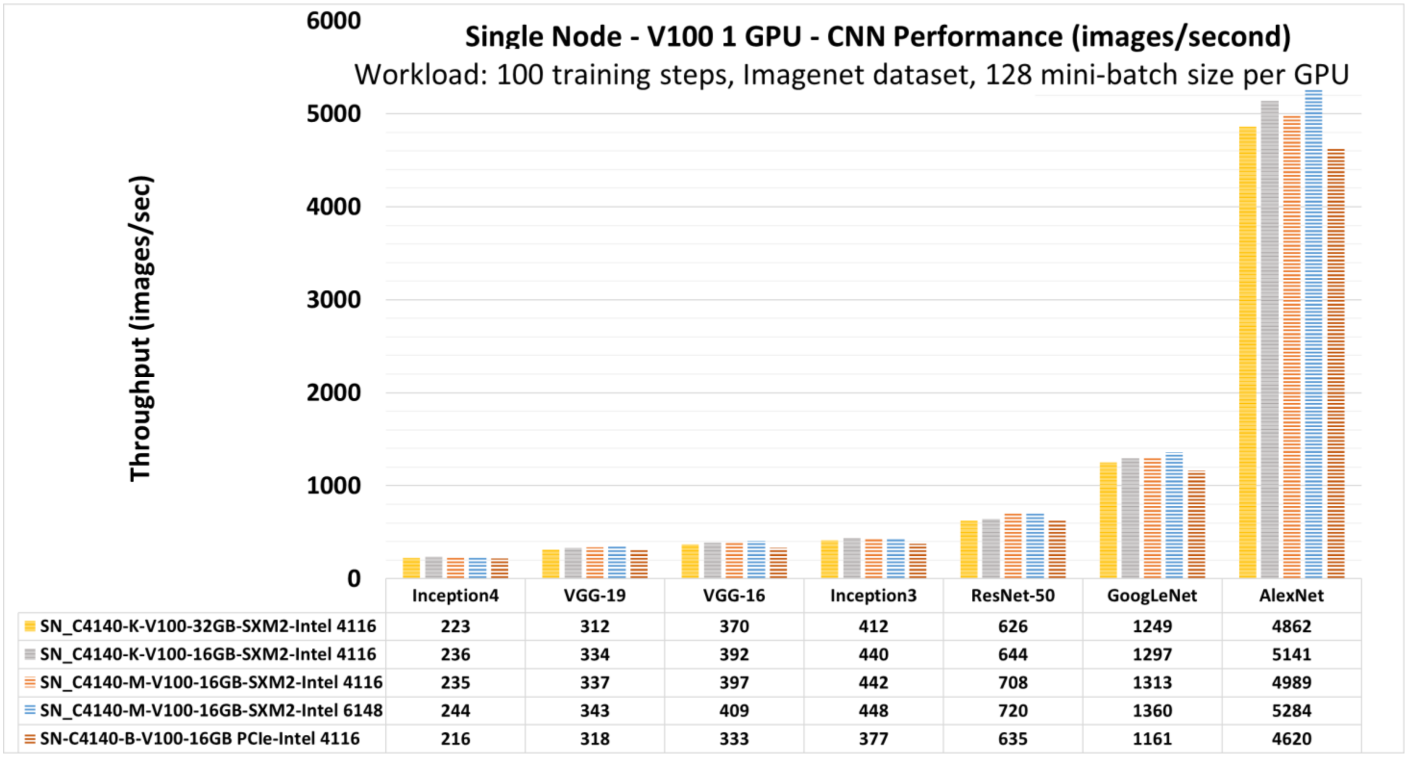


Figure 21 : Performance difference between Intel Xeon-4116 & Intel Xeon-6148 in C4140-M

- As you notice in *Figure 21*, there is a slight performance difference for most of the neural models when using Intel Xeon 6148 except for AlexNet where it shows almost doubling in performance.
- There are quite a few in-depth articles in showing the architecture for AlexNet and at some later point we will try to explain why AlexNet behaves differently based on the number of CPU cores and frequency.

7.1.9 Conclusion

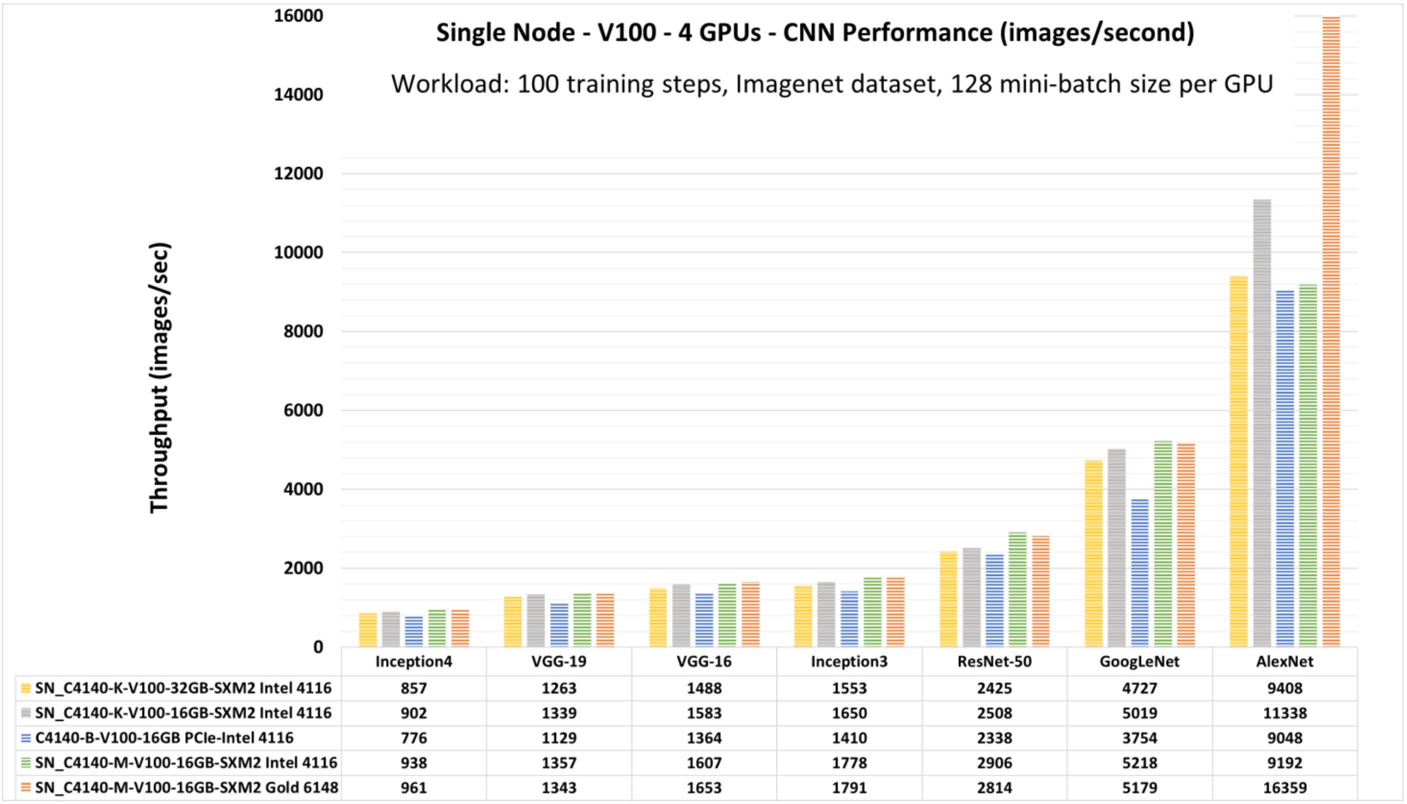
7.1.9.1 Single Node: 1GPU



	Inception4	VGG-19	VGG-16	Inception3	ResNet-50	GoogLeNet	AlexNet
SN_C4140-K-V100-32GB-SXM2-Intel 4116	223	312	370	412	626	1249	4862
SN_C4140-K-V100-16GB-SXM2-Intel 4116	236	334	392	440	644	1297	5141
SN_C4140-M-V100-16GB-SXM2-Intel 4116	235	337	397	442	708	1313	4989
SN_C4140-M-V100-16GB-SXM2-Intel 6148	244	343	409	448	720	1360	5284
SN-C4140-B-V100-16GB PCIe-Intel 4116	216	318	333	377	635	1161	4620

Figure 22 : PowerEdge C4140-B V100-16GB PCIe Vs C4140-K V100 SXM2 – 1GPU

7.1.9.2 Single Node: 4GPU

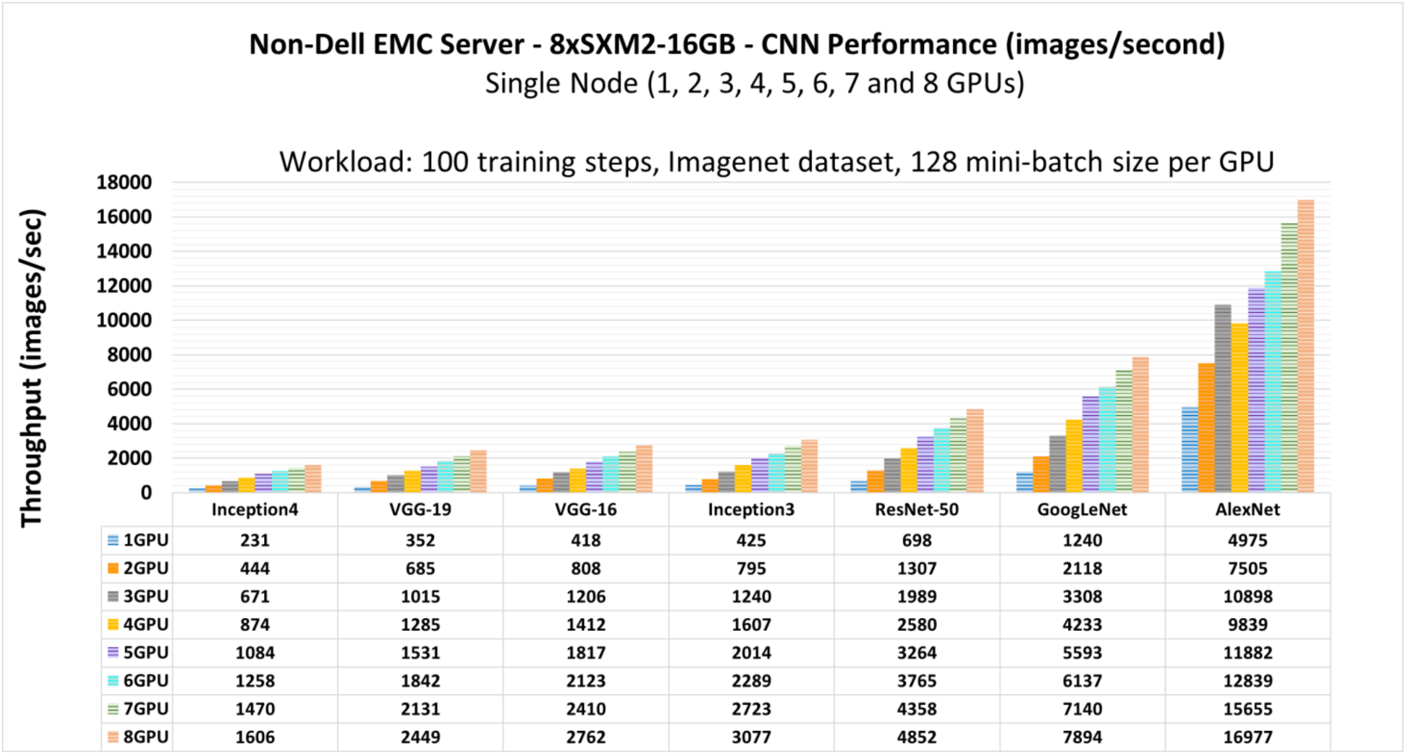


	Inception4	VGG-19	VGG-16	Inception3	ResNet-50	GoogLeNet	AlexNet
SN_C4140-K-V100-32GB-SXM2 Intel 4116	857	1263	1488	1553	2425	4727	9408
SN_C4140-K-V100-16GB-SXM2 Intel 4116	902	1339	1583	1650	2508	5019	11338
C4140-B-V100-16GB PCIe-Intel 4116	776	1129	1364	1410	2338	3754	9048
SN_C4140-M-V100-16GB-SXM2 Intel 4116	938	1357	1607	1778	2906	5218	9192
SN_C4140-M-V100-16GB-SXM2 Gold 6148	961	1343	1653	1791	2814	5179	16359

Figure 23 : PowerEdge C4140-B P100-16GB PCIe vs C4140-K V100-SXM2: 4GPU

- As shown in the plots above in *Figure 22* and *Figure 23* there is not much difference in throughput (images/sec) when comparing PowerEdge C4140-Configuration B with V100-16GB PCIe GPU and Configuration K with V100-16GB SXM2 GPU. The reason is in both configurations GPUs are in peer-peer mode behind PCIe Switch.
- C4140-M shows higher performance compared to C4140-K when using either Intel Xeon CPU with 12 cores or Intel Xeon CPY with 20 cores.

7.1.10 Non-Dell EMC server: 8x V100-16GB-SXM2 – Single Node



	Inception4	VGG-19	VGG-16	Inception3	ResNet-50	GoogLeNet	AlexNet
1GPU	231	352	418	425	698	1240	4975
2GPU	444	685	808	795	1307	2118	7505
3GPU	671	1015	1206	1240	1989	3308	10898
4GPU	874	1285	1412	1607	2580	4233	9839
5GPU	1084	1531	1817	2014	3264	5593	11882
6GPU	1258	1842	2123	2289	3765	6137	12839
7GPU	1470	2131	2410	2723	4358	7140	15655
8GPU	1606	2449	2762	3077	4852	7894	16977

Figure 24: Non-Dell EMC 8x V100-16GB-SXM2 performance

7.2 Throughput images/s – Multi Node

7.2.1 PowerEdge C4130-P100 16GB PCIe- Multi Node

PowerEdge C4130 each with 4 P100-PCIe GPUs were configured in multi-node using InfiniBand RDMA to run the TensorFlow in distributed mode.

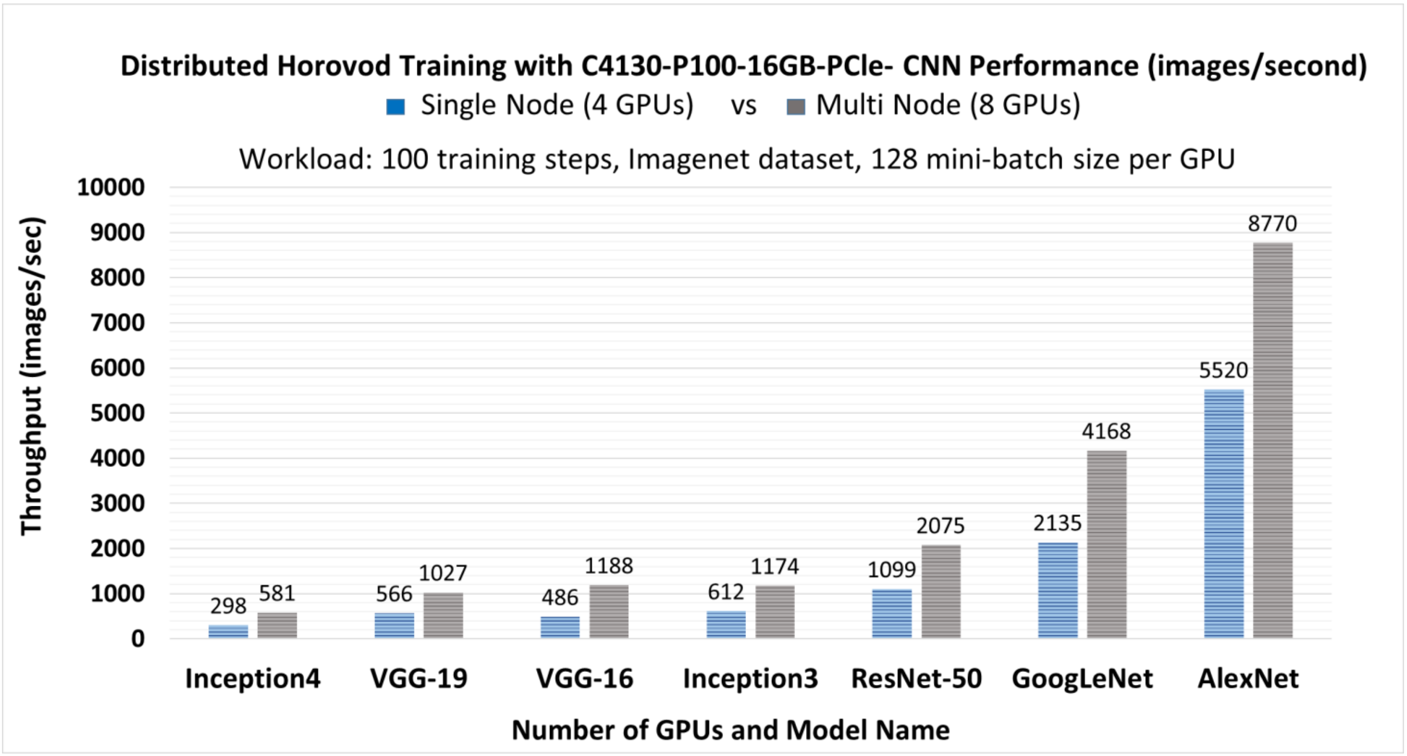
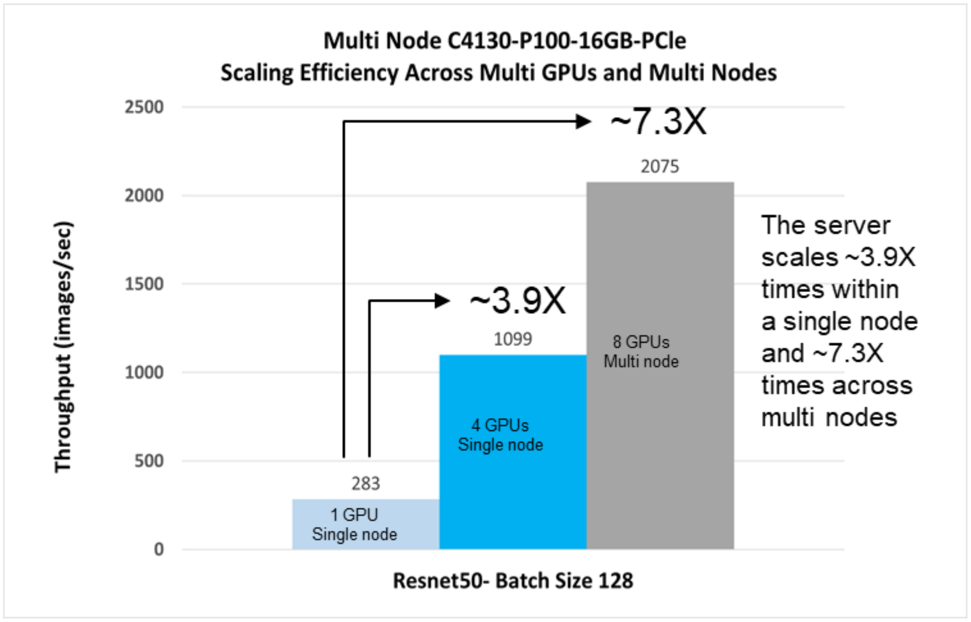
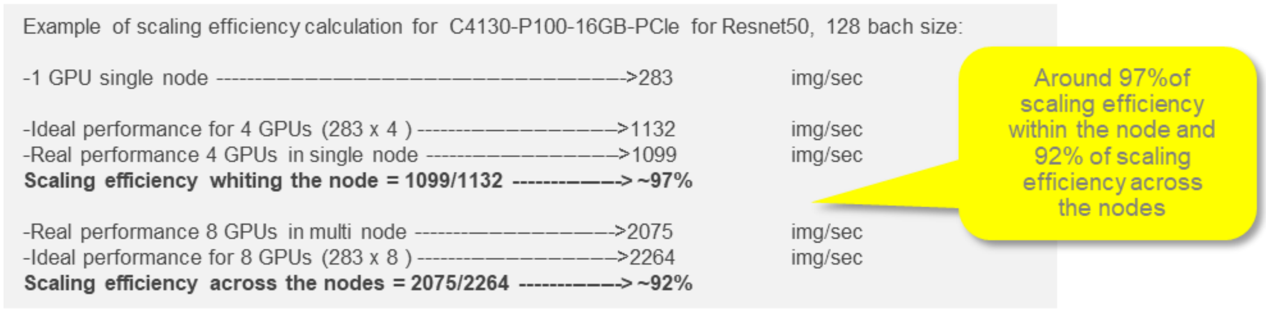


Figure 25: Training with PowerEdge C4130-P100-16GB-PCle in multi-node

PowerEdge C4130 server scales very well within a node with 97% efficiency and 92% across the nodes. The ideal performance is computed by multiplying the single-GPU throughput by the number of GPUs in the system. See *Figure 26*.





Example of scaling efficiency calculation for C4130-P100-16GB-PCIe for Resnet50, 128 bach size:

-1 GPU single node -->283 img/sec

-Ideal performance for 4 GPUs (283 x 4) ------------------->1132 img/sec
-Real performance 4 GPUs in single node ------------------>1099 img/sec
Scaling efficiency whiting the node = 1099/1132 -----------> ~97%

-Real performance 8 GPUs in multi node ------------------->2075 img/sec
-Ideal performance for 8 GPUs (283 x 8) ------------------>2264 img/sec
Scaling efficiency across the nodes = 2075/2264 -----------> ~92%

Around 97%of scaling efficiency within the node and 92% of scaling efficiency across the nodes

Figure 26: Scaling Efficiency of C4130-P100-16GB-PCIe across multi GPUs and multi nodes

7.2.2 PowerEdge C4140-K-V100-16GB and V100-32GB: SXM2 Multi Node

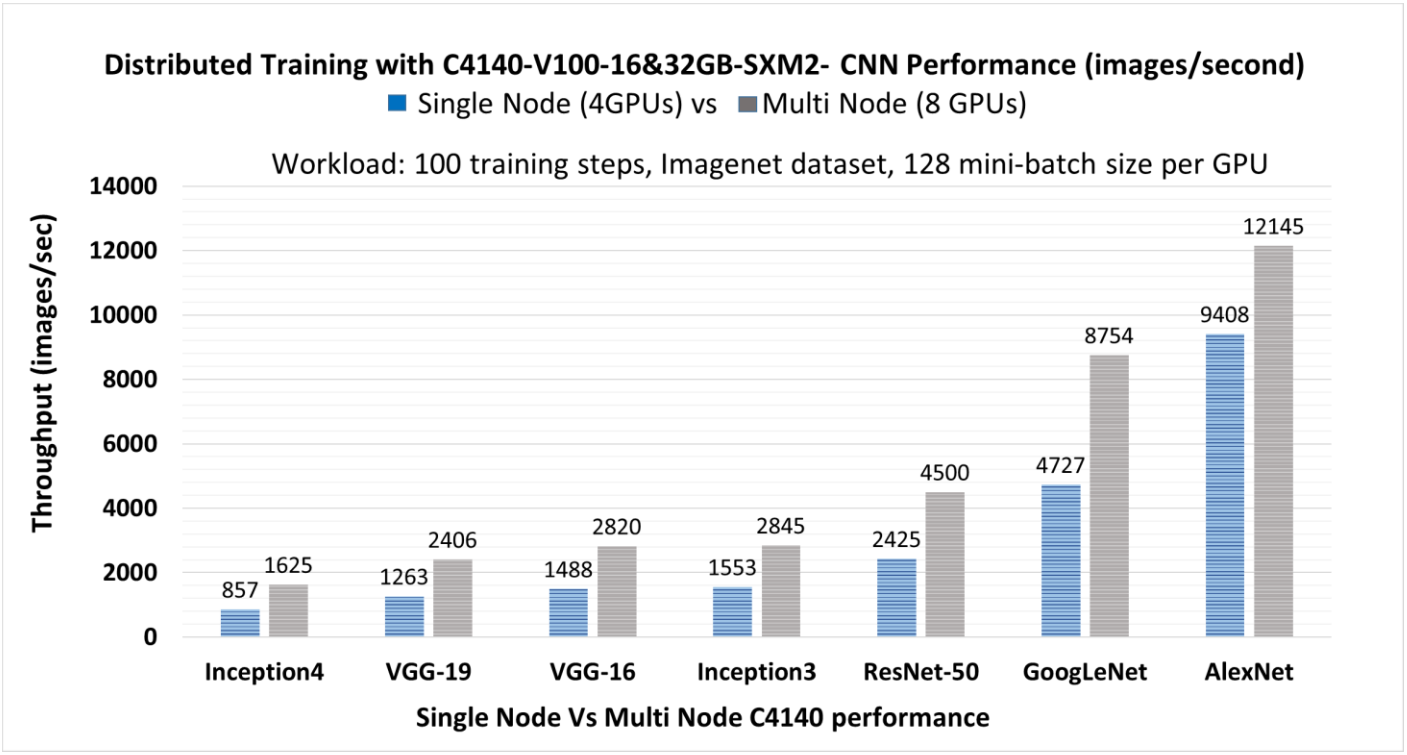
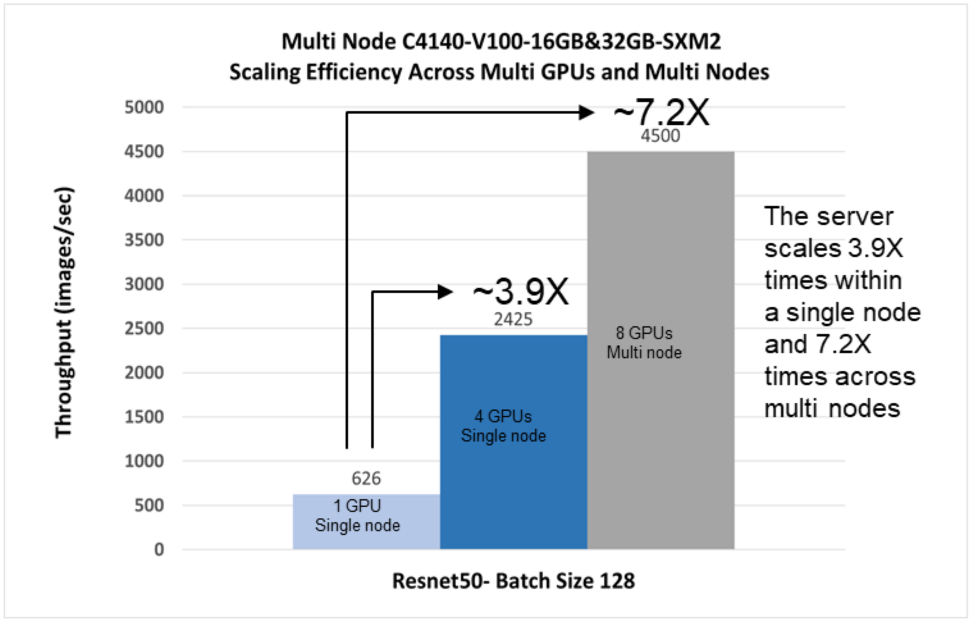
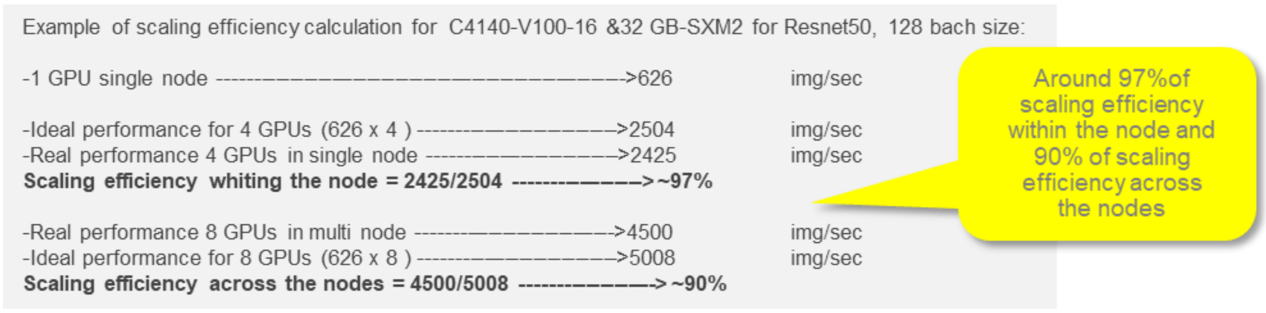


Figure 27: Training with PowerEdge C4140-V100-16&32GB-SXM2 in multi-node

PowerEdge C4140-V100-16GB-SXM2 and PowerEdge C4140-V100-32GB-SXM2 with 4 GPUs each were configured in multi-node to run the TensorFlow in distributed mode, extract the throughput performance, and determine its scaling efficiency. The GPUs scale very well within a node to 97% and 90% across the nodes. The ideal performance is computed by multiplying the single-GPU throughput by the number of GPUs in the system. See *Figure 28*




Example of scaling efficiency calculation for C4140-V100-16 &32 GB-SXM2 for Resnet50, 128 bach size:

-1 GPU single node --->626 img/sec

-Ideal performance for 4 GPUs (626 x 4) --------------------------->2504 img/sec
-Real performance 4 GPUs in single node --------------------------->2425 img/sec
Scaling efficiency whiting the node = 2425/2504 -------------->~97%

-Real performance 8 GPUs in multi node ----------------------------->4500 img/sec
-Ideal performance for 8 GPUs (626 x 8) --------------------------->5008 img/sec
Scaling efficiency across the nodes = 4500/5008 --------------->~90%

Around 97%of scaling efficiency within the node and 90% of scaling efficiency across the nodes

Figure 28: Scaling Efficiency of PowerEdge C4140-V100-16GB-SXM2 and PowerEdge C4140-V100-32GB-SXM2 across multi GPUs and multi nodes

7.2.3 PowerEdge C4140-M-V100-16GB-SXM2 Multi Node

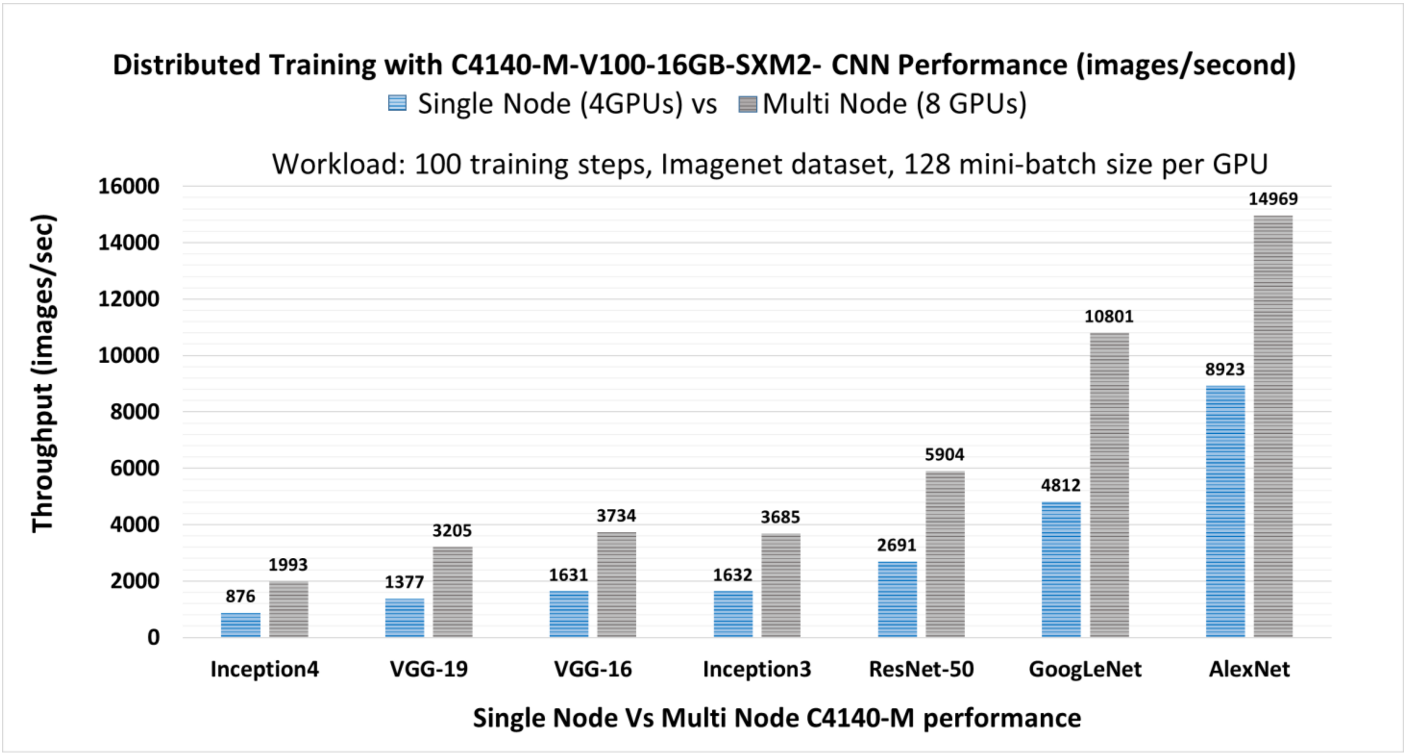
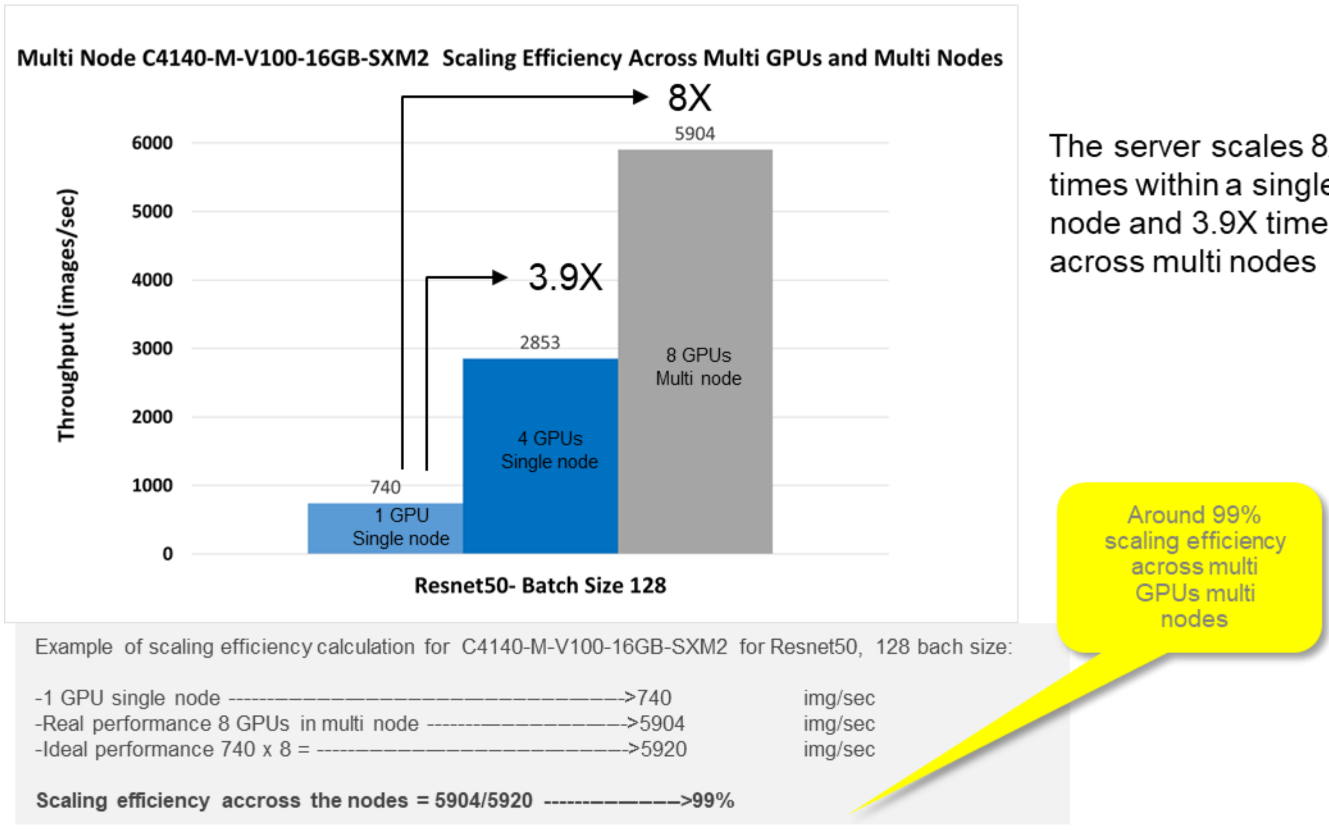


Figure 29. Training with PowerEdge C4140-M-V100-16GB-SXM2 in multi-node



The server scales 8X times within a single node and 3.9X times across multi nodes

Example of scaling efficiency calculation for C4140-M-V100-16GB-SXM2 for Resnet50, 128 bach size:

```
-1 GPU single node --------------------------------->740          img/sec
-Real performance 8 GPUs in multi node ------------------>5904    img/sec
-Ideal performance 740 x 8 = ----------------------->5920         img/sec
```

Scaling efficiency accross the nodes = 5904/5920 ------------->99%

Figure 30. Scaling Efficiency of PowerEdge C4140-M-V100-16GB-SXM2 across multi nodes

7.2.4 PowerEdge C4140-K Multi Node Training vs Non-Dell EMC 8x V100-16GB-SXM2

The Non-Dell EMC 8x V100-16GB- SXM2 system was tested on Nimbix cloud.

Figure 31 shows its throughput performance of 8X SXM2 and shows the comparison versus PowerEdge C4140-K-V100 in distributed mode (8 GPUs).

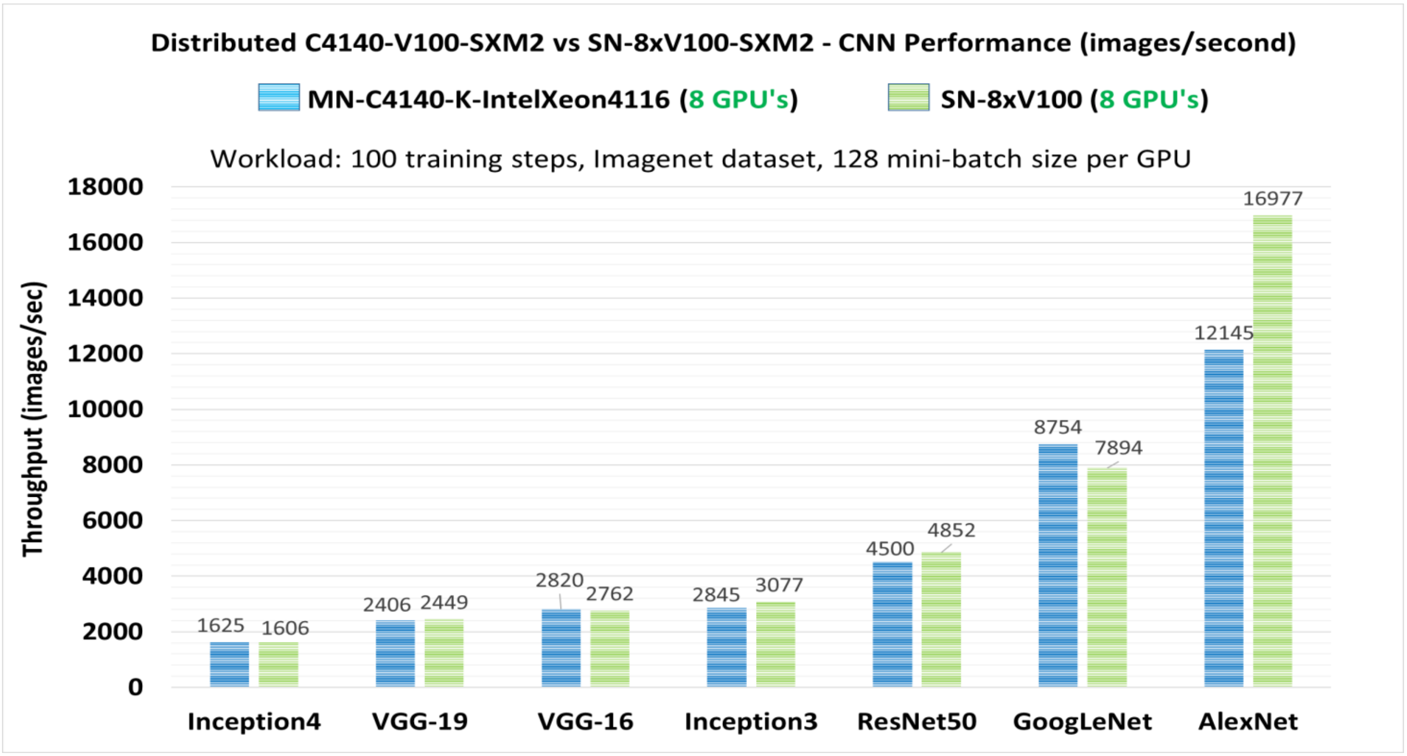


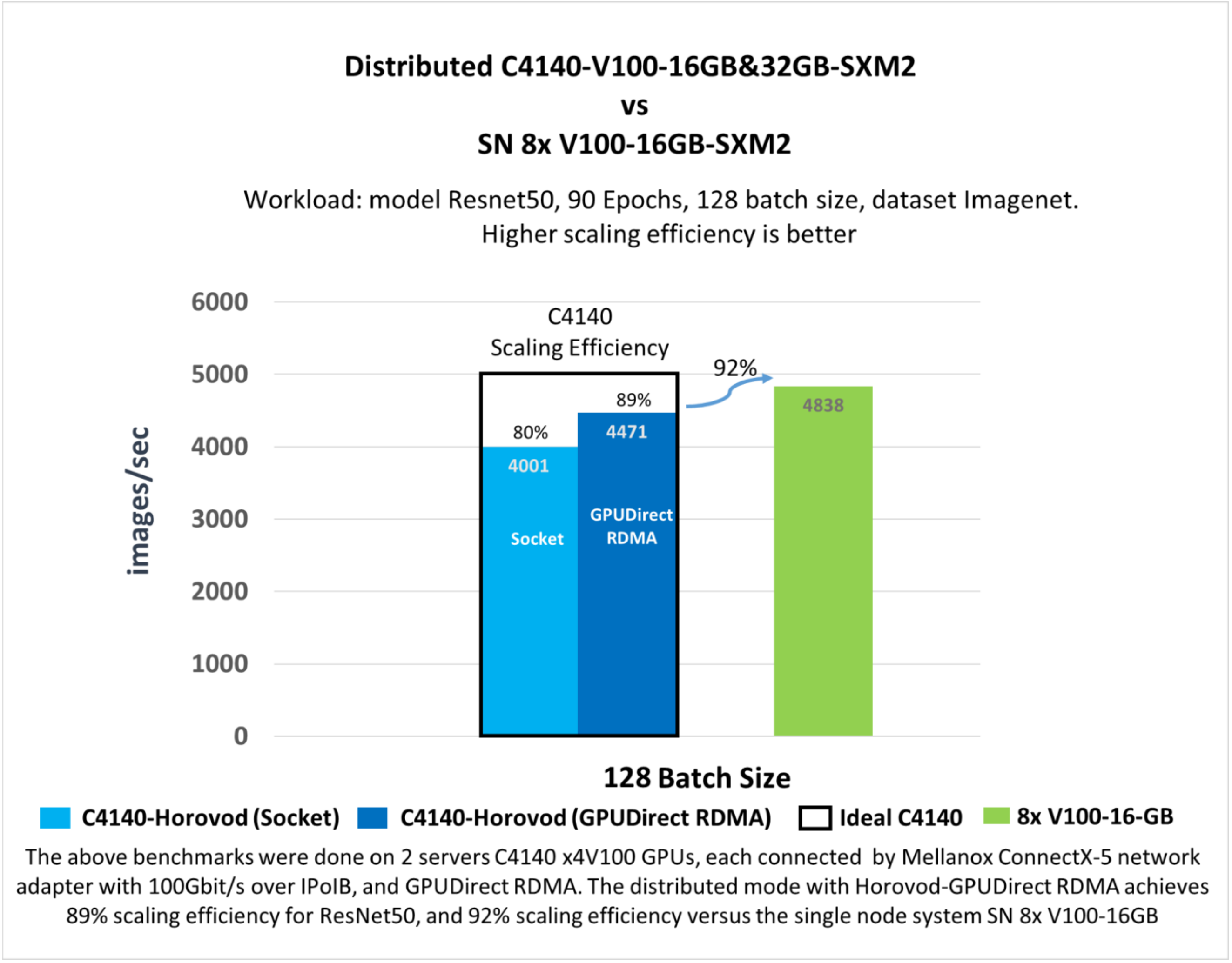
Figure 31: Training with PowerEdge C4140-K-V100-16&32GB-SXM2 (8 GPUs) – multi-node versus Non-Dell EMC SN_8x-V100-16GB-SXM2

	SN_8X V100_16GB- SXM2	MN- PowerEdge C4140-K-V100-SXM2 (16Gb &32GB)-IntelXeon4116	% Diff
Inception-v4	1606	1625	-1.21%
VGG-19	2449	2406	1.78%
VGG-16	2762	2820	-2.03%
Inception-v3	3077	2845	8.16%
ResNet-50	4852	4500	7.81%
GoogLeNet	7894	8754	-9.82%
AlexNet	16977	12145	39.79%

Table 5: 8x GPU Comparison between PowerEdge C4140-K multi-node and 8X SXM2

As seen from the table above, using PowerEdge C4140 with SXM2 shows pretty good performance across various pre-trained neural models. The most common ones i.e. ResNet-50 and Inception-v3 show performance within 8% of 8X SXM2. The only exception is AlexNet where it shows quite a bit of difference between 8X SXM2 and PowerEdge C4140.

The good performance shown by PowerEdge C4140 in multi node mode, comparable to a single node server 8x V100-16GB, was reached after the right software stack configuration with the distributed framework Horovod over IB/GPUDirect-RDMA, see below *Figure 32* the scaling efficiency reached by PowerEdge C4140:



The above benchmarks were done on 2 servers C4140 x4V100 GPUs, each connected by Mellanox ConnectX-5 network adapter with 100Gbit/s over IPoIB, and GPUDirect RDMA. The distributed mode with Horovod-GPUDirect RDMA achieves 89% scaling efficiency for ResNet50, and 92% scaling efficiency versus the single node system SN 8x V100-16GB

Figure 32: The Performance with Distributed Horovod TensorFlow, connected by Mellanox ConnectX-5 network adapter with 100Gbit/s over IPoIB, and GPUDirect RDMA

7.2.5 PowerEdge C4140-M Multi Node Training vs Non-Dell EMC 8x V100-16GB-SXM2

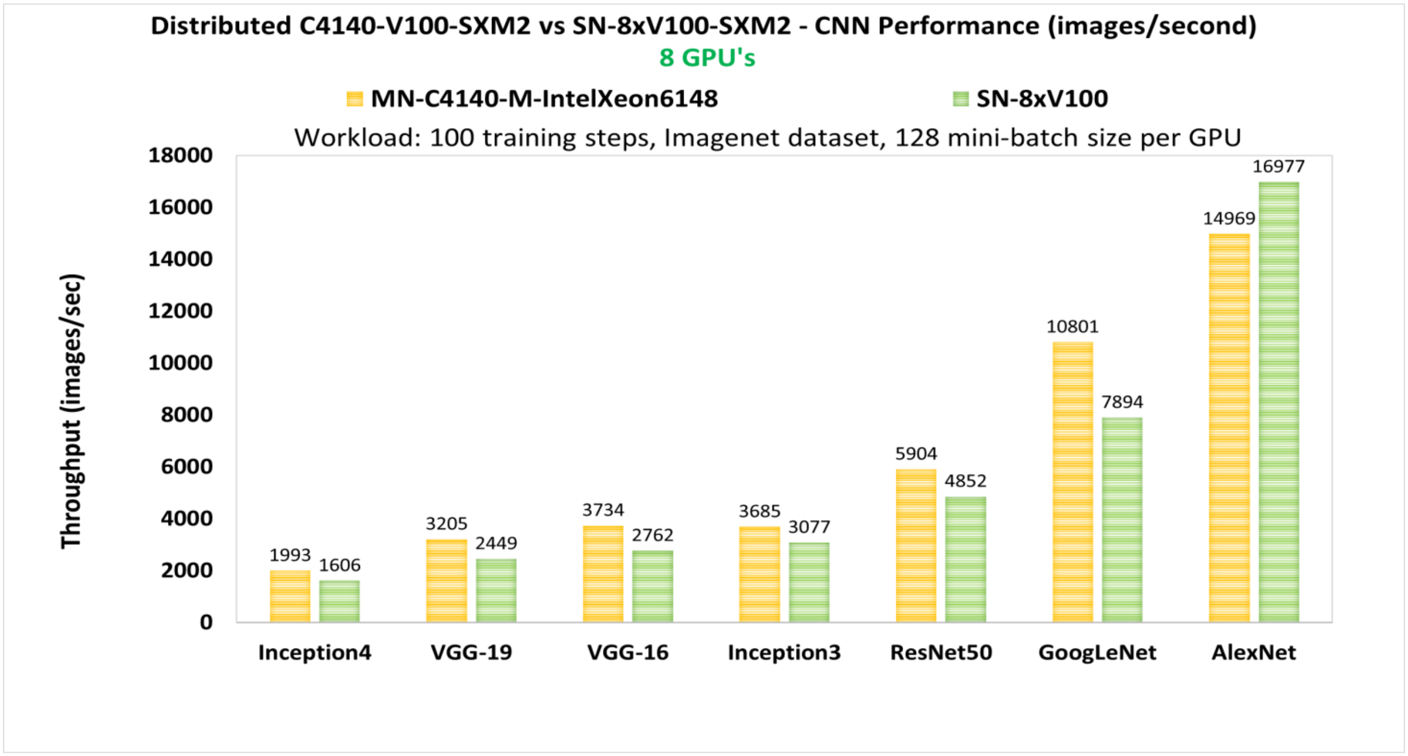


Figure 33. Training with PowerEdge C4140-M-V100-16GB-SXM2 (8 GPUs) – multi-node versus Non-Dell EMC SN_8x-V100-16GB-SXM2

In the *Figure 33* above we can appreciate the throughput improvement when using a sever with a higher capacity CPU; as seen in the table below, almost all the models trained with C4140-M-V100-16GB-SXM2 - CPU IntelXeon6148 (8 GPUs) – multi-node performed better than SN-8xV100. The exception was AlexNet which still performed under SN_8xV100; however, it improved its throughput significantly compared when trained with the server with C4140-K-V100-16GB-SXM2 - IntelXeon4116. See the summary in the below table

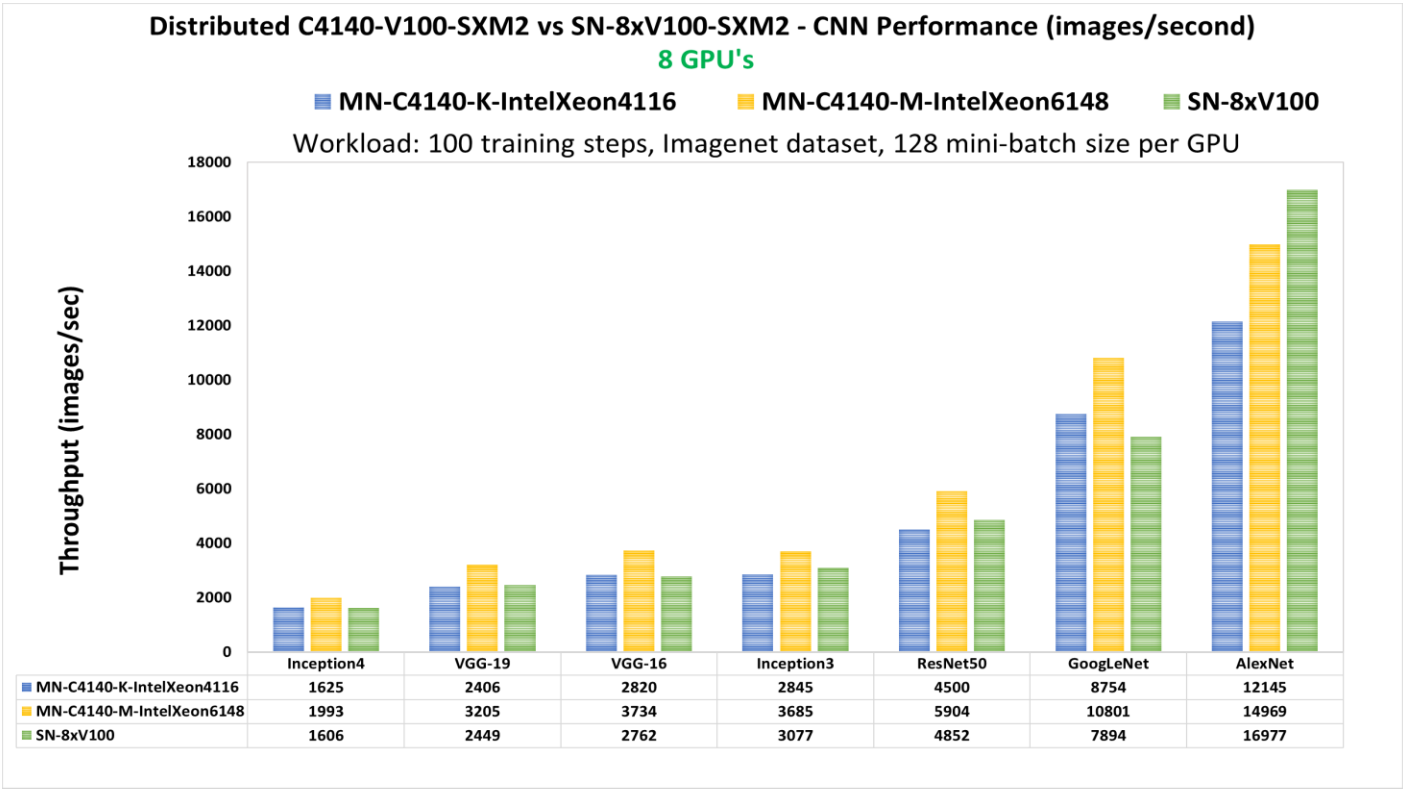
	SN_8X V100_16GB- SXM2	**MN- PowerEdge C4140-M-V100-SXM2 16GB**	**% Diff**
Inception-v4	1606	1993	19%
VGG-19	2449	3205	24%
VGG-16	2762	3734	26%
Inception-v3	3077	3685	16%
ResNet-50	4852	5904	18%
GoogLeNet	7894	10801	27%
AlexNet	16977	14969	-13%

Table 6: Table 5: 8x GPU Comparison between PowerEdge C4140-M multi-node and 8X SXM2

7.2.6 PowerEdge C4140 Multi Node Training with Different CPU Models vs 8x V100-16GB-SXM2

In the results shown in the *Figure 31 and Figure 33* we configured the multi-node system with servers PowerEdge C4140-V100-SXM2- Configuration-K Intel Xeon4116 CPU and Configuration-M Intel Xeon6148 CPU respectively, versus single-node training non-Dell EMC 8xV100-16GB-SXM2.

To show the impact of the CPU in the training of deep learning workloads, we run additional tests configuring the multi-node system with servers PowerEdge C4140-V100-SXM2 Configuration-M and Intel Xeon6148 CPU. In the *Figure 34* we see how advance CPU models boost even more the gpu performance, since most of the data loading, data preprocessing, and batch transformation tasks occur at the CPU level, whereas the training tasks occur at the gpu level.



	Inception4	VGG-19	VGG-16	Inception3	ResNet50	GoogLeNet	AlexNet
▦ MN-C4140-K-IntelXeon4116	1625	2406	2820	2845	4500	8754	12145
▦ MN-C4140-M-IntelXeon6148	1993	3205	3734	3685	5904	10801	14969
▦ SN-8xV100	1606	2449	2762	3077	4852	7894	16977

Figure 34 . Multi-node training PowerEdge C4140-V100-SXM2- Configuration-K with IntelXeon4116 cpu, Multi-node training PowerEdge C4140-V100-SXM2 Configuration-M with IntelXeon6148 cpu, versus single-node training non Dell 8xV100-16GB-SXM2

7.3 Results Showing Training Time Vs Accuracy

These tests were run with 90 epochs to determine training time to achieve top-1% and top-5% accuracy.

Figure 35 shows the results for the server 8X SXM2, POWEREDGE C4140-V100 in single and multi-node mode, C4130-P100 in single and multi-node mode, and R740-P40.

Results Highlights

- The fastest training time was achieved by the system 8X SXM2 with 93% of accuracy convergence in 6.6 hours.
- PowerEdge C4140 – Configuration K with SXM2 in multi-node configuration achieved 95% of accuracy convergence in 7.2 hours.
- R740-P40 with 48.8hrs and 93% of accuracy convergence.

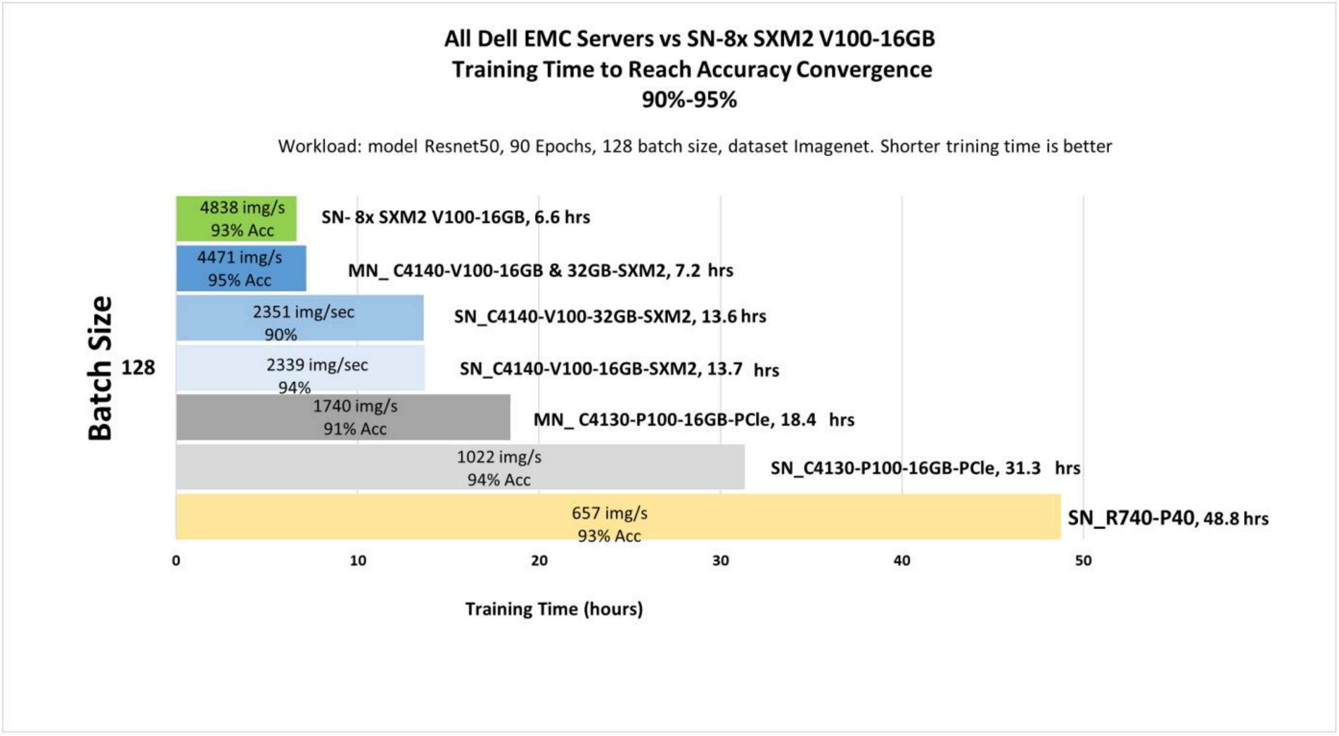


Figure 35: Longest tests to extract accuracy convergence and training time

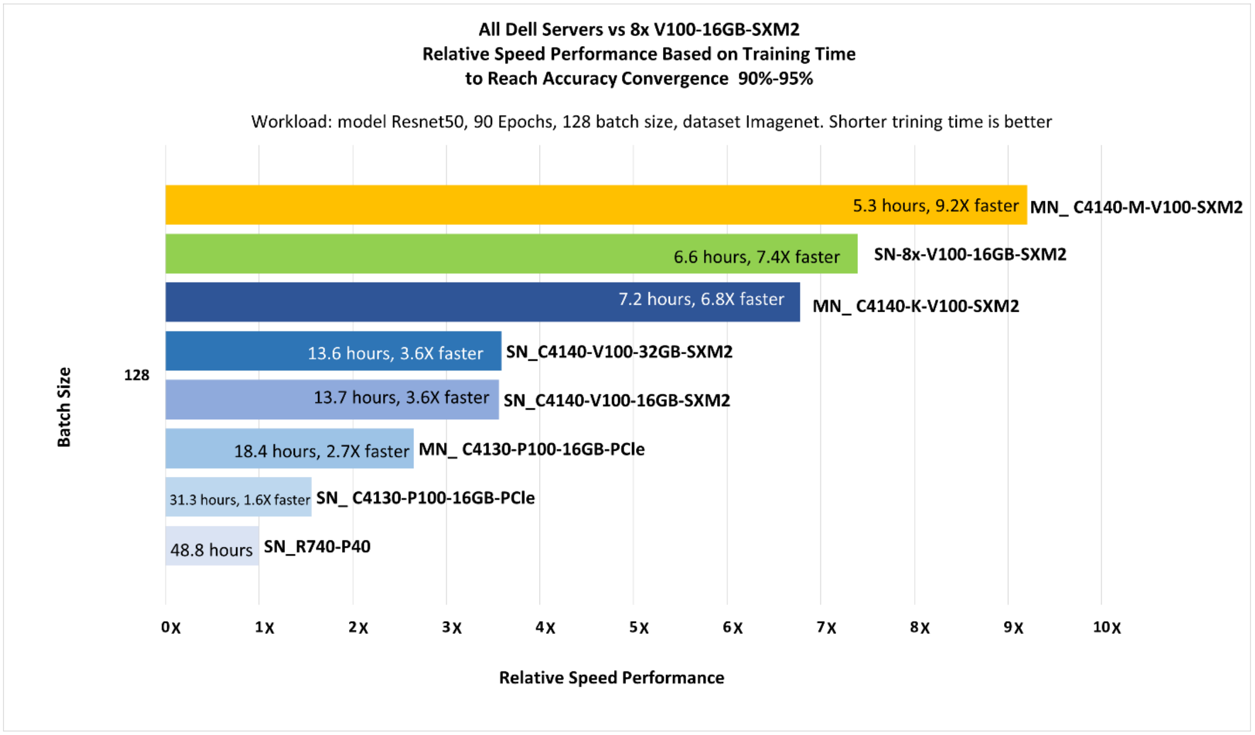


Figure 36: Relative speed performance based on training time

After training the PowerEdge C4140 – Configuration M with SXM2 in multi-node configuration, we saw it reached the fastest training in 5.3 hours, overpassing the Non-Dell EMC SN_8x-V100-16GB-SXM2 which completed the training time in 6.6 hours. See *Figure 36*

7.3.1 Elapsed Training Time for Several Models

Another aspect we wanted to explore was the accuracy convergence capacity for other models, so we selected models with different depth network topology (vg199, ResNet50, and Inception-v4) and ran the long tests on PowerEdge C4140 in multi-node configuration and non-Dell EMC 8x – V100 SXM2. The results are show in *Figure 37* below.

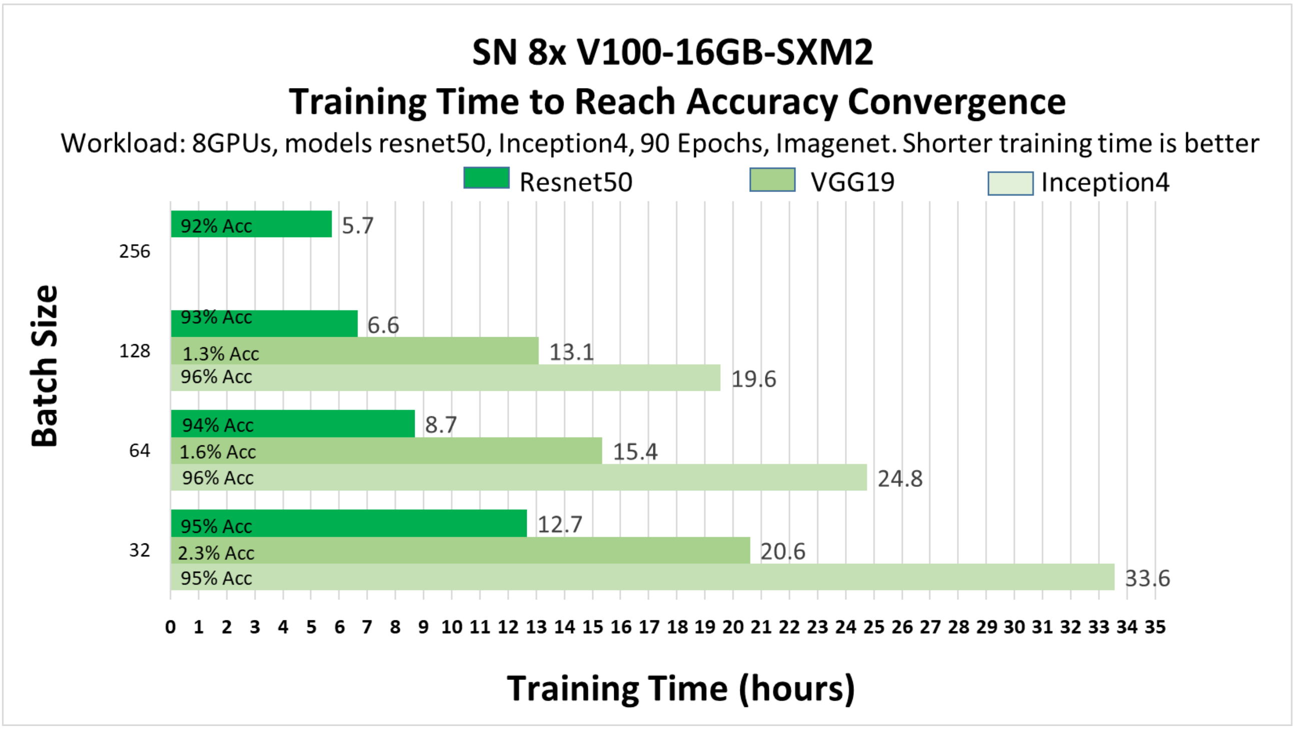


Figure 37: Training long tests to extract accuracy convergence and training time with 8X SXM2 and different models

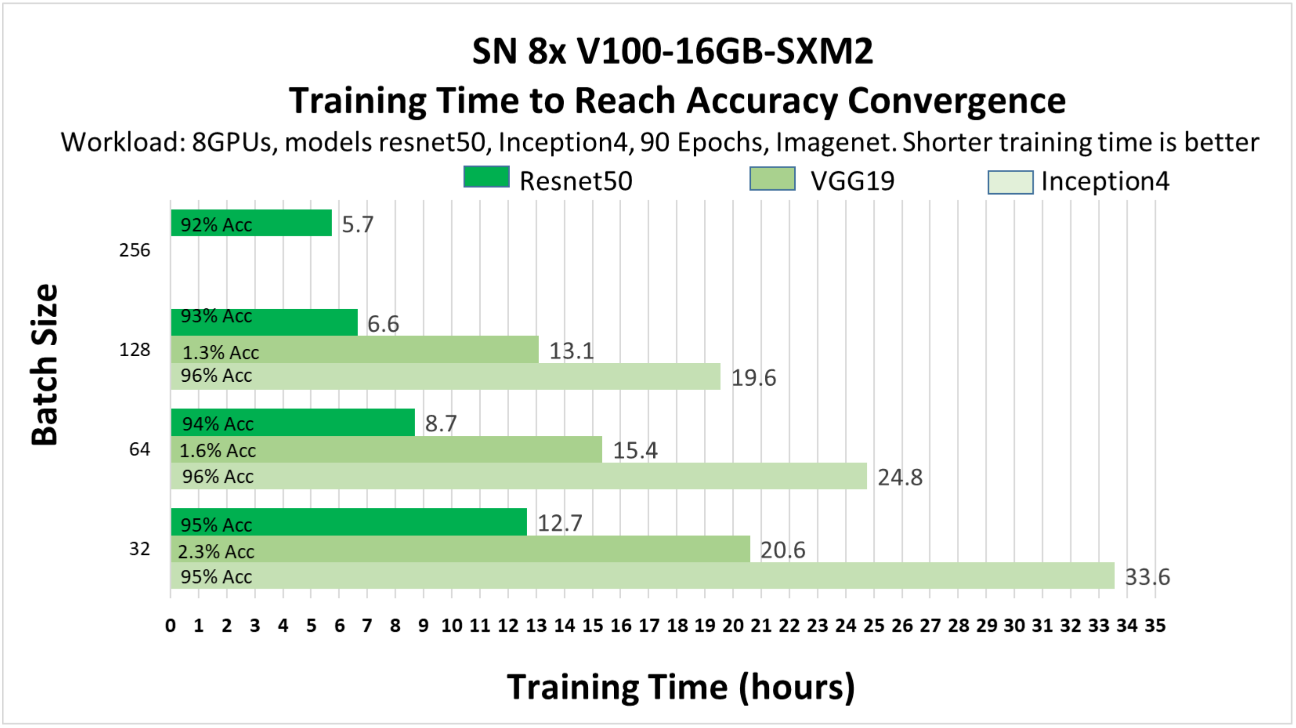


Figure 37 above we observe that the models ResNet50 and Inception-v4 reached between 92%-96% of accuracy convergence in 90 epochs; however, ResNet50 with different batch sizes

converged faster than Inception-v4. On the other hand, the model VGG-19 didn't produce acceptable accuracy suggesting it requires over 90 epochs to converge.

Figure 38 below show comparable results for PowerEdge C4140-K (multi-node) – V100 SXM2

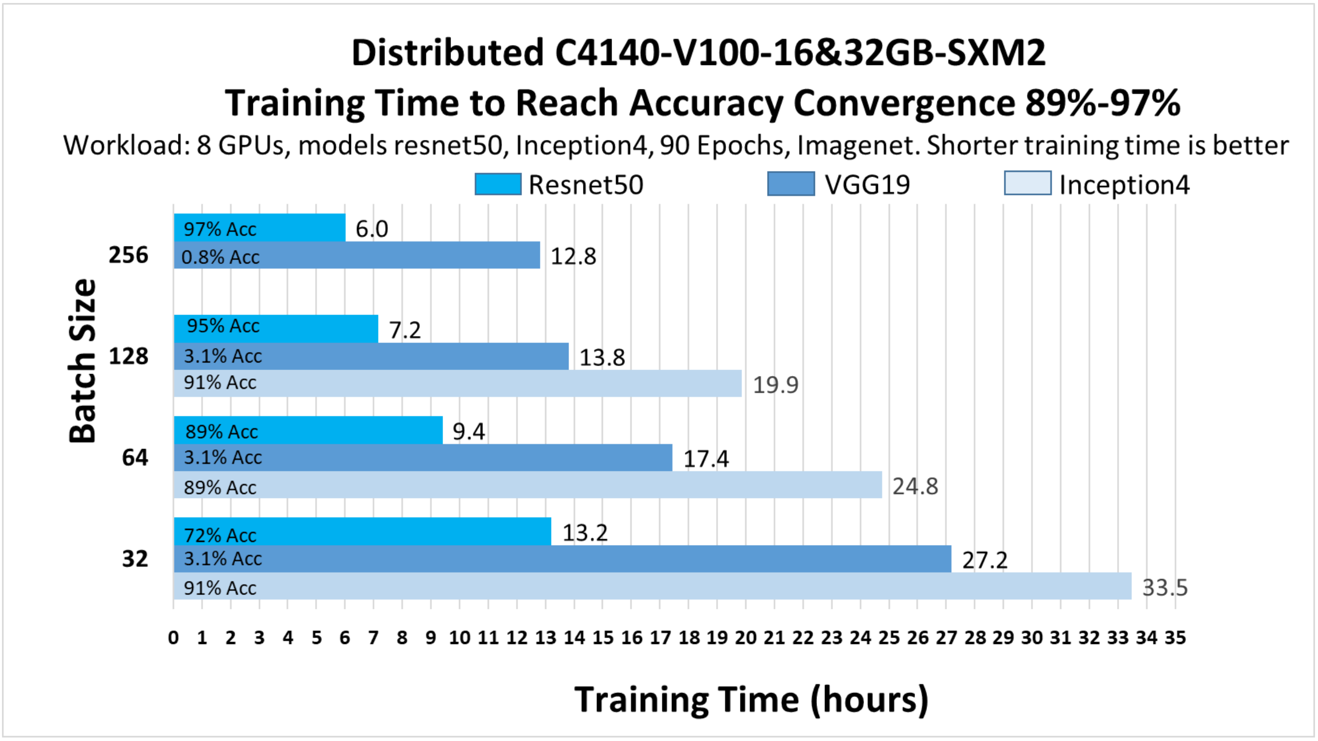


Figure 38: Training long tests to extract accuracy convergence and training time with PowerEdge C4140-K multi-node and different models

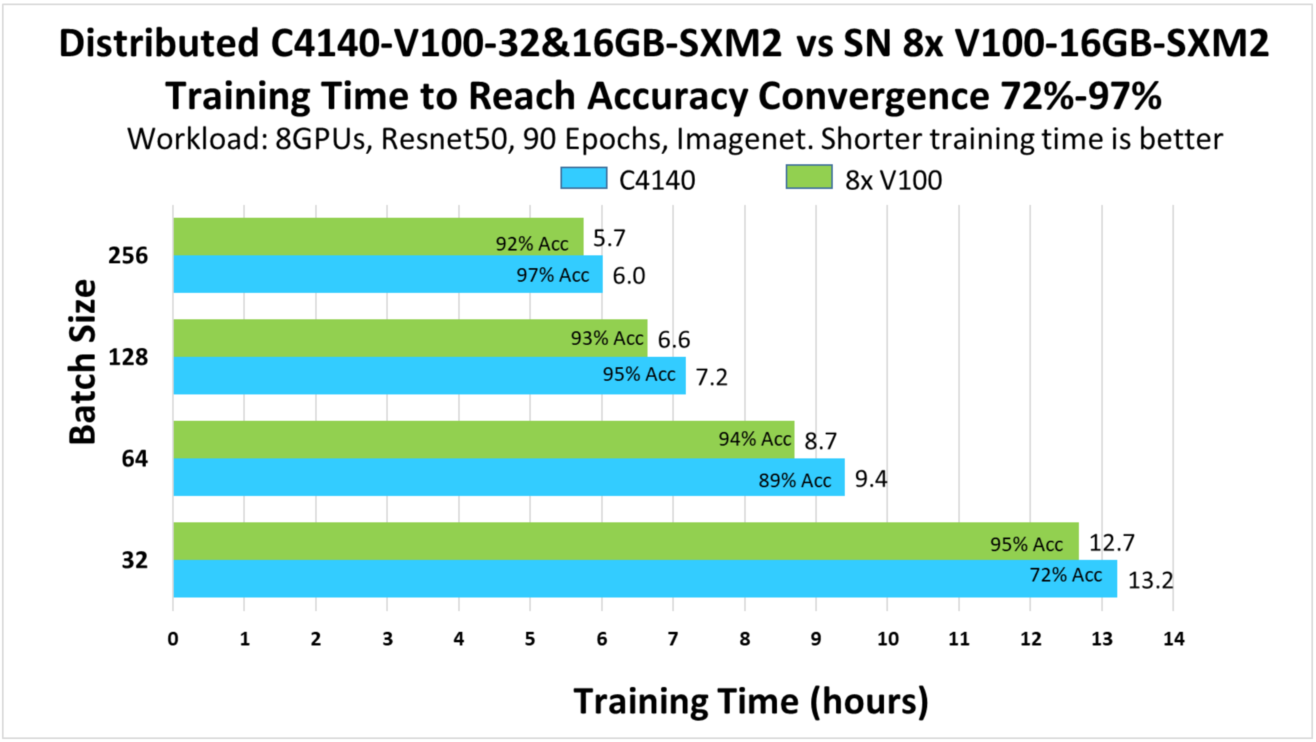


Figure 39: Training long tests to extract accuracy convergence and training time with PowerEdge C4140-K multi-node and single-node 8x V100-SXM2 with different models

Figure 39 above shows comparison between 8X SXM2 and PowerEdge C4140 Configuration-K in multi-node configuration using ResNet-50. The training time difference between 8X SXM2 and multi-node PowerEdge C4140 is within 7% which shows that using Mellanox InfiniBand RDMA allows PowerEdge C4140 to achieve similar performance as a scale-up server.

To show the impact of the CPU in the training of deep learning workloads to reach the accuracy convergence, we run additional tests configuring the multi-node system with servers PowerEdge C4140-V100-SXM2 Configuration-M and IntelXeon6148 CPU. In the *Figure 40* we see the multi-node system C4140-V100-SXM2 Configuration-M and IntelXeon6148 CPU performs **1.3X** faster than SN-8xV100 for the model resnet50 trained in several batch sizes. Again, it shows the relationship between the CPU model and the Deep Learning performance, where most of the data loading, data preprocessing, and batch transformation tasks occur at the CPU level, whereas the training tasks occur at the gpu level.

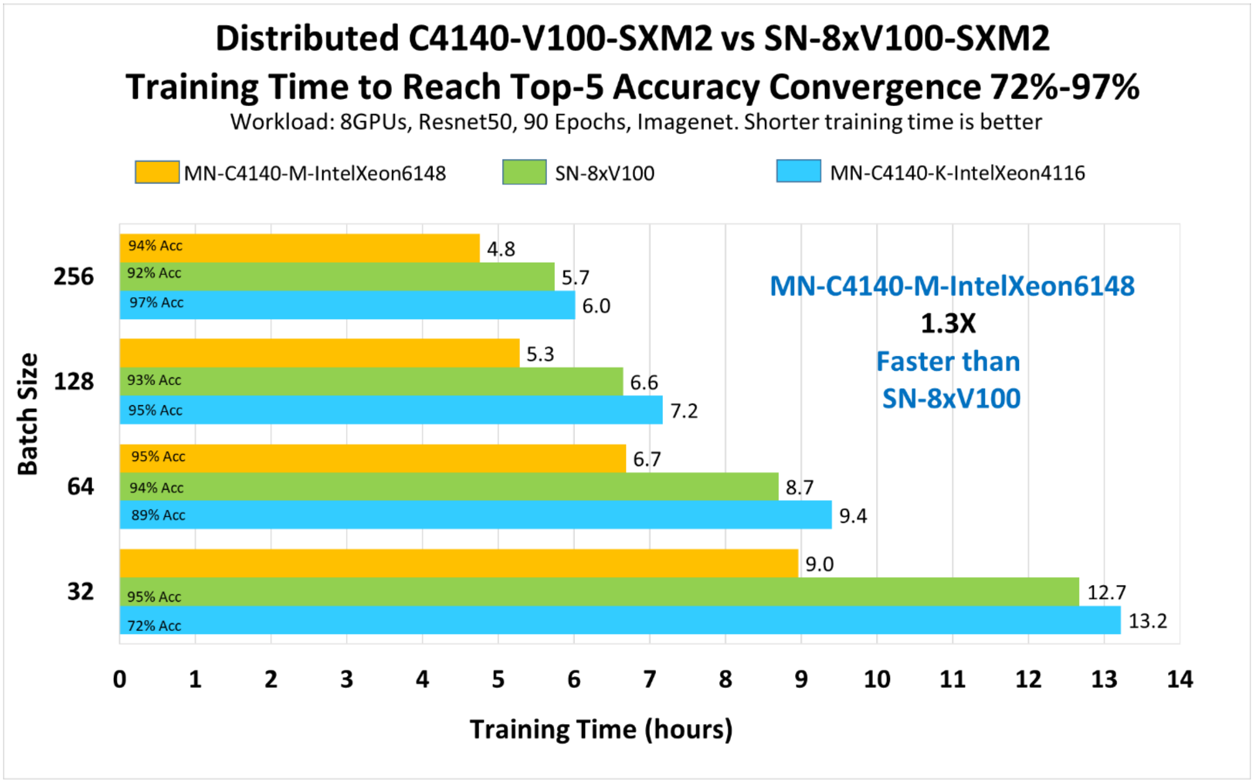


Figure 40. Multi-node training PowerEdge C4140-V100-SXM2- Configuration-K with IntelXeon4116 cpu, Multi-node training PowerEdge C4140-V100-SXM2 Configuration-M with IntelXeon6148 cpu, versus single-node training non Dell 8xV100-16GB-SXM2

In the Figure *40* we can see how the system C4140-V100-SXM2 Configuration-M outperforms in terms of training time in different batch sizes compared the other systems.

7.4 Other Explored Aspects

This section shows the results of aspects explored during this project such as the hyper parameter tuning, learning rate effect on single-node and multi-node mode, and critical kernels executed in the TensorFlow benchmarks. These aspects could be subject of deeper study for future projects.

7.4.1 Hyper-parameters tuning

The section below are the commands with the hyper-parameter tuning used to maximize the throughput performance in single and distributed mode server implementations.

Figure 41 shows the high impact of the hyper-parameter tuning in the throughput performance:

Single Node – TensorFlow:

```
#python3 tf_cnn_benchmarks.py --variable_update=replicated --
data_dir=/data/imagenet_tfrecord/train --data_name=imagenet --model=ResNet50 --batch_size=128 --
device=gpu --num_gpus=4 --num_epochs=90 --print_training_accuracy=true --summary_verbosity=0 --
momentum=0.9 --piecewise_learning_rate_schedule='0.4;10;0.04;60;0.004' --weight_decay=0.0001 --
optimizer=momentum --use_fp16=True --local_parameter_device=gpu --all_reduce_spec=nccl --
display_every=1000
```

Distributed Horovod – TensorFlow:

```
#mpirun -np 8 -H 192.168.11.1:4,192.168.11.2:4 -x NCCL_IB_DISABLE=0 -x NCCL_IB_CUDA_SUPPORT=1 -
x NCCL_SOCKET_IFNAME=ib0 -x NCCL_DEBUG=INFO --bind-to none --map-by slot --mca plm_rsh_args "-
p 50000" python tf_cnn_benchmarks.py --variable_update=horovod --
data_dir=/data/imagenet_tfrecord/train --data_name=imagenet --model=ResNet50 --batch_size=128 --
num_epochs=90 --display_every=1000 --device=gpu --print_training_accuracy=true --
summary_verbosity=0 --momentum=0.9 --piecewise_learning_rate_schedule='0.4;10;0.04;60;0.004' --
weight_decay=0.0001 --optimizer=momentum --use_fp16=True --local_parameter_device=gpu --
horovod_device=gpu --datasets_num_private_threads=4
```

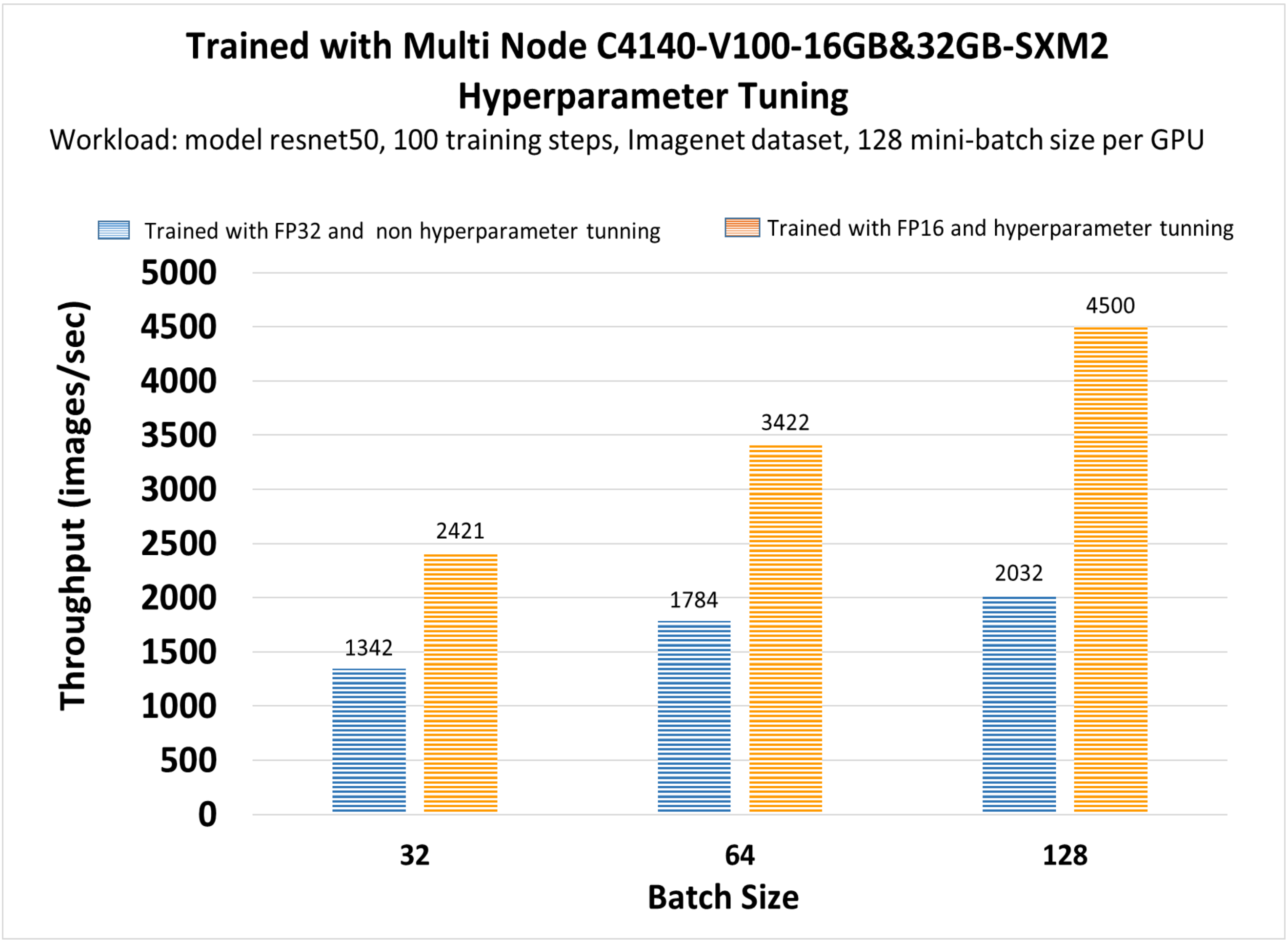


Figure 41: Effect of the hyper-parameter tuning in the throughput performance

7.4.2 Learning Rate Effect in Distributed Mode

In this experiment, we used the learning rate schedule as follows: the initial learning rate was set up to 0.4 for the first 10 epochs, after that the learning rate was decreased to 0.04 until the model reached 60 epochs of training, finally it was decreased to 0.004 until the end of the training with 90 epochs.

Figure 42 & *Figure 43* we show how learning rate has a different effect in single node versus multi node. In multi node mode the learning rate update is not reflected immediately and there is a delay until the change is reflected in every processor.

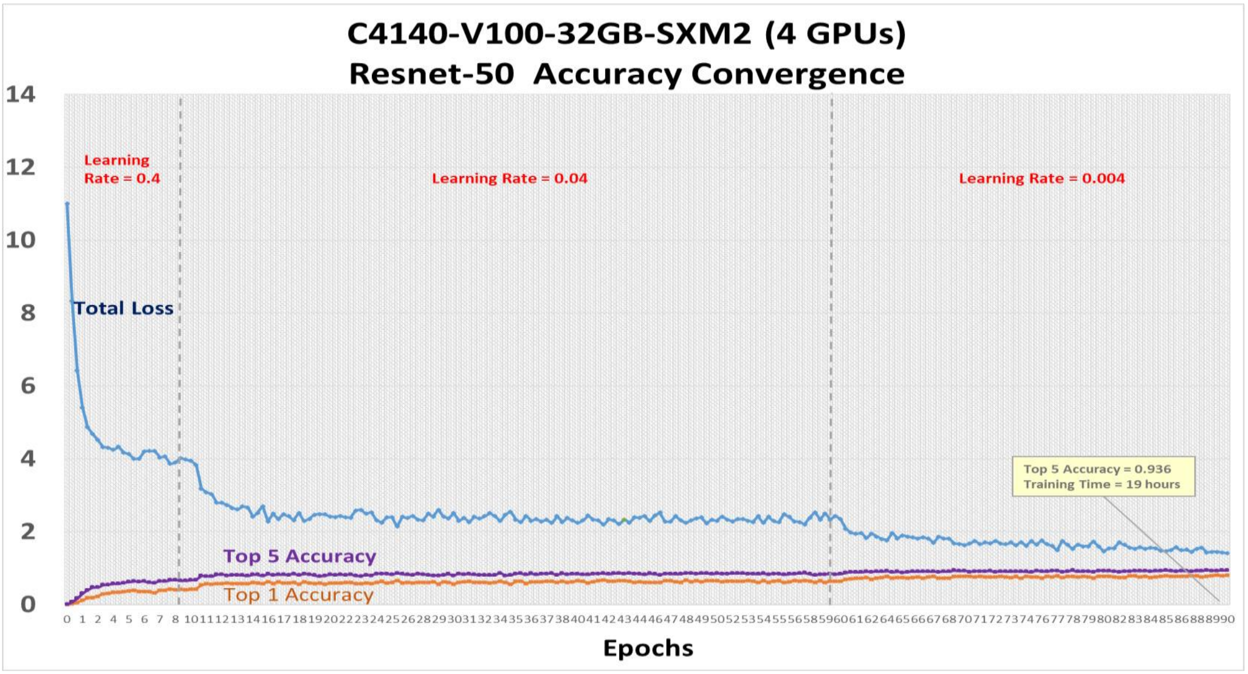


Figure 42: Training with PowerEdge C4140-V100-16&32GB-SXM2 (4 GPUs) – single-node

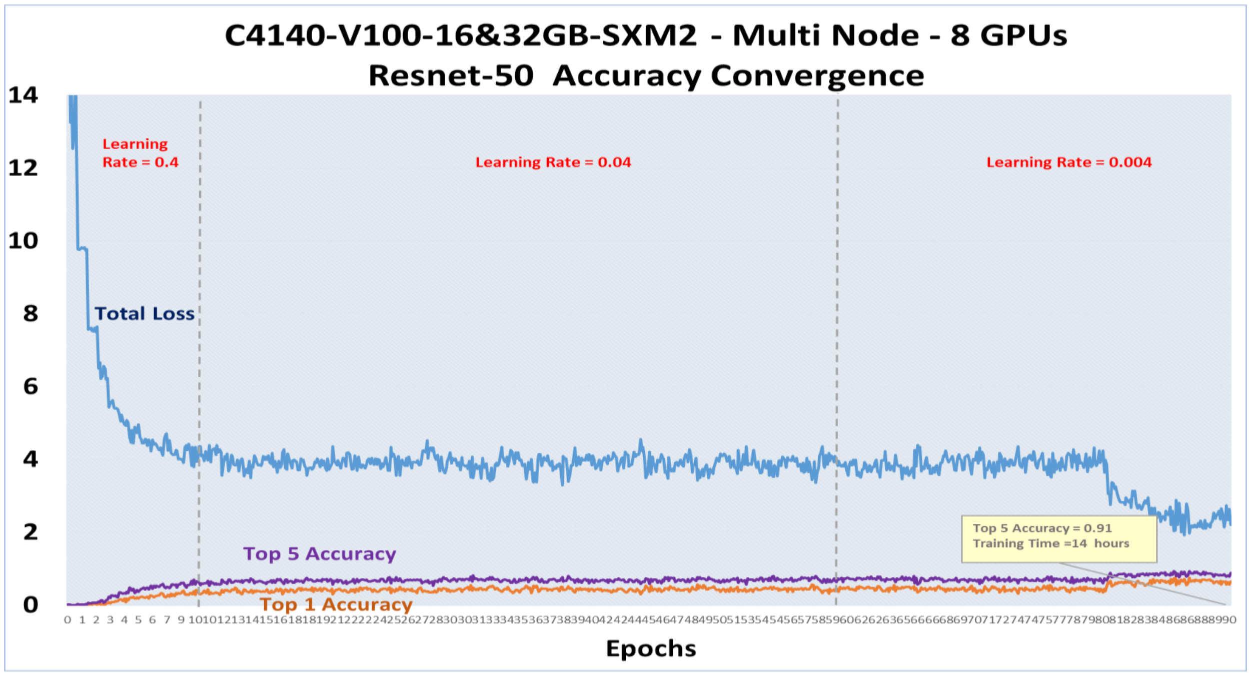


Figure 43: Training with PowerEdge C4140-V100-16&32GB-SXM2 (8 GPUs) – multi-node

7.4.3 Communication and Neural Networks Primitives

We wanted to explore the critical kernels executed in one GPU when running the TensorFlow benchmarks, so we used the Nvidia profiling tool Nvprof to analyze one TensorFlow benchmark trained with C4130-P100-PCIe-16GB in multi-node.

Figure 44 shows the critical kernels executed, we found that the communication primitives all reduce where called 38.4% of the time, it may suggest that the GPU may spend too much time exchanging and communicating rather than computing, this is something that can be explored in depth for future projects.

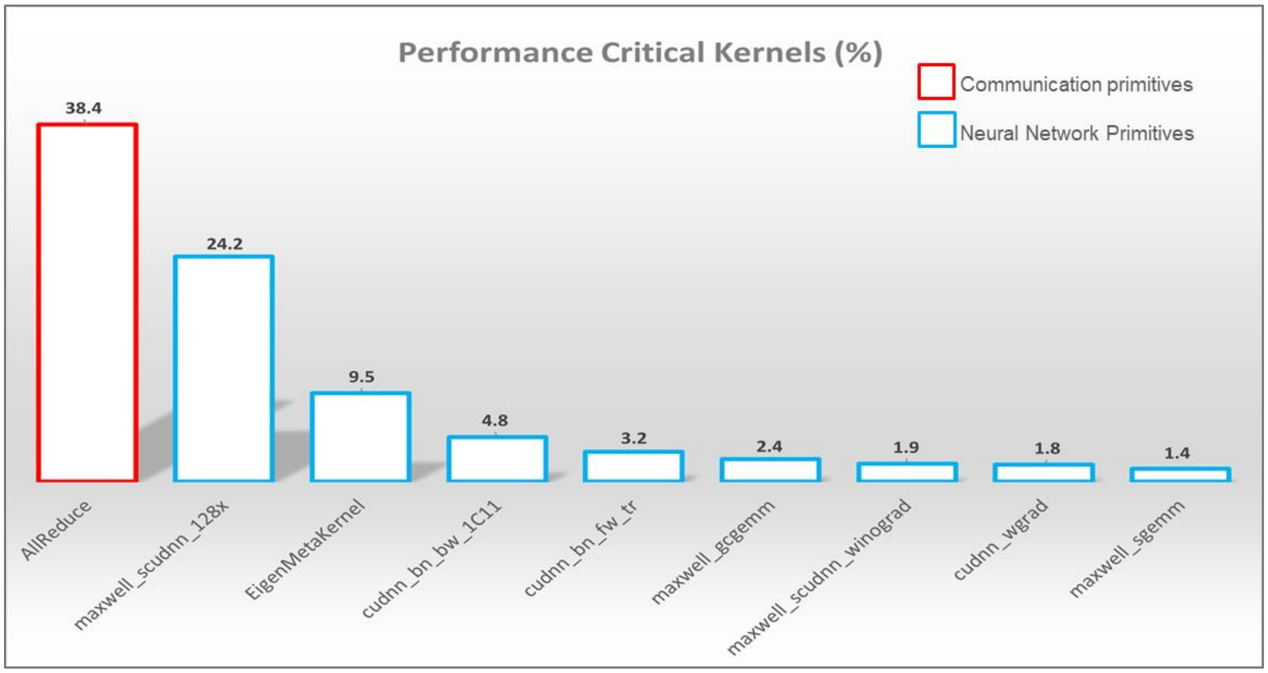


Figure 44: Critical kernels executed when training with C4130-P100-16GB-SXM2 (8 GPUs) – multi-node

CUDA Toolkit offers several performance analysis tools to optimize the performance of CUDA or OpenACC applications. The Visual Profiler nvvp traces CUDA activities, profiles CUDA kernels, and correlates performance instrumentation with source code. The tool Nvprof collects performance events and metrics; CUDA memcheck detects memory accesses issues, incorrect GPU thread synchronization and other important aspects to optimize performance [5].

8 Conclusion and Future Work

- PowerEdge C4140 using Nvidia 4x NVLink architecture scales relatively well when using Uber Horovod distributed training library and Mellanox InfiniBand RDMA as the high-speed link between nodes.

- Table 5 shows that PowerEdge C4140 in multi-node configuration for most widely used model ResNet-50 is within 7.8% of single node Non-Dell EMC 8x-NVLink system. But with C4140-M in multi-node out performs single node 8x NVLink by at least 18% using ResNet-50. The only disclaimer is that C4140-M results are using the latest version of NCCL & TensorFlow containers.

- There is 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.

- Some of our future work in this area will be related to exploring the latest software optimizations being released by Nvidia and looking at fast.ai library where Jeremy Howard and researchers at fast.ai achieved training time of 3 hours on 8x V100 on ResNet-50.

9 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}
}
```

10 References

- [1] Nvidia Blogs, "What's the Difference between Artificial Intelligence, Machine Learning, and Deep Learning?" [Online]. Available: https://blogs.Nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

- [2] Cornell University Library, "Horovod: fast and easy distributed deep learning in TensorFlow" [Online]. Available: https://arxiv.org/abs/1802.05799

- [3] 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

- [4] Horovod GitHub, "Horovod in Docker" [Online]. Available: https://github.com/uber/horovod/blob/master/docs/docker.md

- [5] Nvidia, "CUDA Toolkit Documentation" [Online], https://docs.Nvidia.com/cuda/profiler-users-guide/index.html#profiling-overview

- [6] Cornell University Library, "Training ImageNet in 1 Hour" [Online]. Available: https://arxiv.org/abs/1706.02677

- [7] Medium, "Hardware for Deep Learning. Part 3: GPU" [Online]. Available: https://blog.inten.to/hardware-for-deep-learning-part-3-gpu-8906c1644664

- [8] Sergeev, A., Del Balso, M. (2017) *Meet Horovod: Uber's Open Source Distributed Deep Learning Framework for TensorFlow*. Retrieved from https://eng.uber.com/horovod/

- [9] Sergeev, A. (2017) *Horovod - Distributed TensorFlow Made Easy*. Retrieved from https://www.slideshare.net/AlexanderSergeev4/horovod-distributed-tensorflow-made-easy

- [10] Sergeev, A., Del Balso, M. (2018) *Horovod: fast and easy distributed deep learning in TensorFlow*. arXiv:1802.05799