Deep Learning Performance with P100 GPUs

Authors: Rengan Xu and Nishanth Dandapanthu. Dell EMC HPC Innovation Lab. October 2016

Introduction to Deep Learning and P100 GPU

Deep Learning (DL), an area of Machine Learning, has achieved significant progress in recent years. Its application area includes pattern recognition, image classification, Natural Language Processing (NLP), autonomous driving and so on. Deep learning attempts to learn multiple levels of features of the input large data sets with multi-layer neural networks and make predictive decision for the new data. This indicates two phases in deep learning: first, the neural network is trained with large number of input data; second, the trained neural network is used to test/inference/predict the new data. Due to the large number of parameters (the weight matrix connecting neurons in different layers and the bias in each layer, etc.) and training set size, the training phase requires tremendous amounts of computation power.

To approach this problem, we utilize accelerators which include GPU, FPGA and DSP and so on. This blog focuses on GPU accelerator. GPU is a massively parallel architecture that employs thousands of small but efficient cores to accelerate the computational intensive tasks. Especially, NVIDIA® Tesla® P100™ GPU uses the new Pascal™ architecture to deliver very high performance for HPC and hyperscale workloads. In PCIe-based servers, P100 delivers around 4.7 and 9.3 TeraFLOPS of double and single precision performance, respectively. And in NVLink™-optimized servers, P100 delivers around 5.3 and 10.6 TeraFLOPS of double and single precision performance, respectively. This blog focuses on P100 for PCIe-based servers. P100 is also equipped with High Bandwidth Memory 2 (HBM2) which offers higher bandwidth than the traditional GDDR5 memory. Therefore, the high compute capability and high memory bandwidth make GPU an ideal candidate to accelerate deep learning applications.

Deep Learning Frameworks and Dataset

In this blog, we will present the performance and scalability of P100 GPUs with different deep learning frameworks on a cluster. Three deep learning frameworks were chosen: NVIDIA's fork of Caffe (NV-Caffe), MXNet and TensorFlow. Caffe is a well-known and widely used deep learning framework which is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors. It focuses more on the image classification and it supports multiple GPUs within a node but not across nodes. MXNet, jointly developed by collaborators from multiple universities and companies, is a lightweight, portable and flexible deep learning framework designed for both efficiency and flexibility. This framework scales to multiple GPUs within a node and across nodes. TensorFlow, developed by Google's Brain team, is a library for numerical computation using data flow graphs. TensorFlow also supports multiples GPUs and can scale to multiple nodes.

All of the three deep learning frameworks we chose are able to perform the image classification task. With this in mind, we chose the well-known ImageNet Large Scale Visual Recognition Competition (ILSVRC) 2012 dataset. This training dataset contains 1281167 training images and 50000 validation images. All images are grouped into 1000 categories or classes. Another reason we chose ILSVRC 2012 dataset is that

its workload is large enough to perform long time training and it is a benchmark dataset used by many deep learning researchers.

Testing Methodology

This blog quantifies the performance of deep learning frameworks using NVIDIA's P100-PCIe GPU and Dell's PowerEdge C4130 server architecture. Figure 1 shows the testing cluster. The cluster includes one head node which is Dell's PowerEdge R630 and four compute nodes which are Dell's PowerEdge C4130. All nodes are connected by an InfiniBand network and they share disk storage through NFS. Each compute node has 2 CPUs and 4 P100-PCIe GPUs. All of the four compute nodes have the same configurations. Table 1 shows the detailed information about the hardware configuration and software used in every compute node.

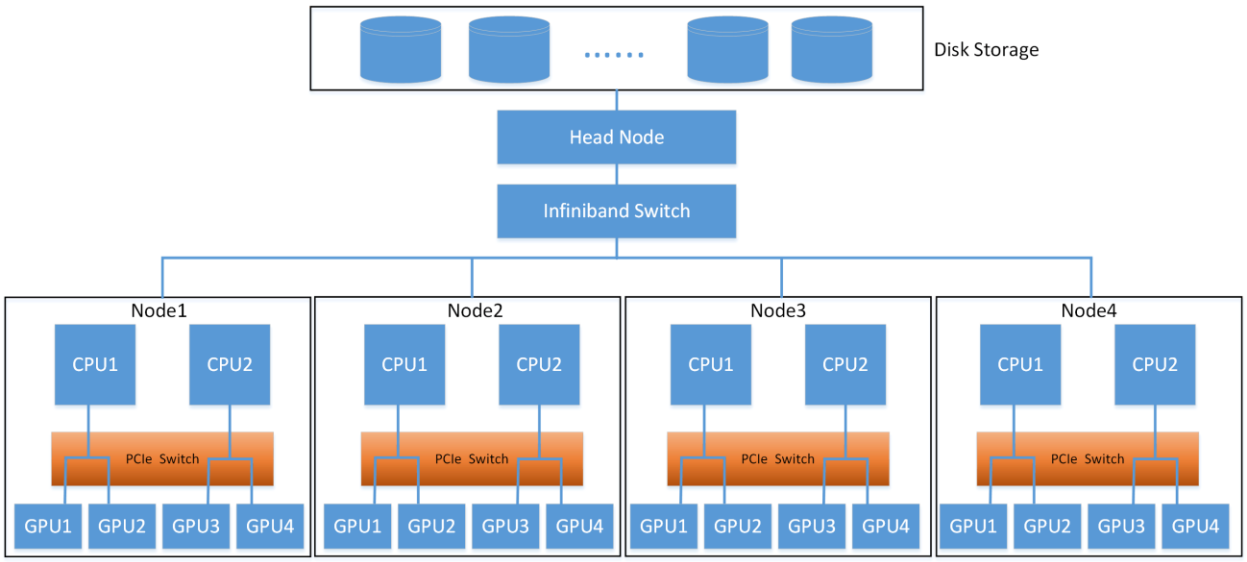

Figure 1: Testing Cluster for Deep Learning

Table 1: Hardware Configuration and Software Details

Platform	PowerEdge C4130 (configuration G)
Processor	2 x Intel Xeon CPU E5-2690 v4 @2.6GHz (Broadwell)
Memory	256GB DDR4 @ 2400MHz
Disk	9TB HDD
GPU	P100-PCIe with 16GB GPU memory
Nodes Interconnects	Mellanox ConnectX-4 VPI (EDR 100Gb/s Infiniband)
Infiniband Switch	Mellanox SB7890
Software and Firmware	
Operating System	RHEL 7.2 x86_64
Linux Kernel Version	3.10.0-327.el7
BIOS	Version 2.1.6
CUDA version and driver	CUDA 8.0 (361.77)
NCCL version	Version 1.2.3

cuDNN library	Version 5.1.3
Intel Compiler	Version 2017.0.098
Python	2.7.5
Deep Learning Frameworks	
NV-Caffe	Version 0.15.13
Intel-Caffe	Version 1.0.0-rc3
MXNet	Version 0.7.0
TensorFlow	Version 0.11.0-rc2

We measured the metrics of both images/sec and training time.

The images/sec is the measurement for training speed while the training time is the wall clock time for training, I/O operation and other overhead. The images/sec number was obtained from "samples/sec" in MXNet and TensorFlow in the output log files. NV-Caffe listed "M s/N iter" as output which means M seconds were taken to process N iterations, or N batches. The metric images/sec was calculated by "batch_size*N/M". The batch size is the number of training samples in one forward/backward pass through all layers of a neural network. The images/sec number was averaged across all iterations to take into account the deviations.

The training time was obtained from "Time cost" in MXNet output logs. For NV-Caffe and TensorFlow, their output log files contained the wall-clock timestamps during the whole training. So the time difference from the start to the end of the training was calculated as the training time.

Since NV-Caffe did not support distributed training, it was not executed on multiple nodes. The MXNet framework was able to run on multiple nodes. However, the caveat was that it could only use the Ethernet interface (10 Gb/s) on the compute nodes by default, and therefore the performance was not as high as expected. To solve this issue, we have manually changed its source code so that the high-speed InfiniBand interface (EDR 100 Gb/s) was used. The training with TensorFlow on multiple nodes was able to run but with poor performance and the reason is still under investigation.

Table 2 shows the input parameters used in different deep learning frameworks. In all deep learning frameworks, the neural network training requires many epochs or iterations. Whether the term epoch or iteration is used is determined by each framework. An epoch is a complete pass through all samples in a given dataset, while one iteration processes only one batch of samples. Therefore, the relationship between iterations and epochs is: epochs = (iterations*batch_size)/training_samples. Every framework only needs either epochs or iterations so that another parameter can be easily determined by this formula. Since our goal was to measure the performance and scalability of Dell's server and not to train an end-to-end image classification model, the training was a subset of the full model training which was large enough to reflect performance. Therefore we chose a smaller number of epochs or iterations so that they could finish in a reasonable time. Although only partial training was performed, the training speed (images/sec) remained relatively constant over this period.

The batch size is one of the hyperparameters the user needs to tune when training a neural network model with mini-batch Stochastic Gradient Descent (SGD). The batch size in the table are commonly used sizes. Whether these batch sizes are optimized for model accuracy is left in future work. For all neural networks in all frameworks, we increased the batch size proportionally with increasing number of GPUs. In the meantime, the number of iterations was adjusted so that the total number of samples was fixed no matter how many GPUs were used. Since epoch has nothing to do with batch size, its value was not changed when a different number of GPUs was used. For MXNet GoogleNet, there was runtime error if different bath sizes were used for different number of GPUs, so we used constant batch size. Learning rate is another hyperparameter that needs to be tuned. In this experiment, the default value in each framework was used.

Table 2: Input parameters used in different deep learning frameworks

	Batch size		Image shape	Iterations/Epochs
NV-Caffe GoogleNet	CPU	128	224	4000 iterations
	1 P100	128		4000 iterations
	2 P100	256		2000 iterations
	4 P100	512		1000 iterations
TensorFlow Inception-V3	1 P100	64	299	4000 iterations
	2 P100	128		2000 iterations
	4 P100	256		1000 iterations
MXNet GoogleNet	1-16 P100	144	256	1 epoch
MXNet Inception-BN	1 P100	64	224	1 epoch
	2 P100	128		
	4 P100	256		
	8 P100	256		
	12 P100	256		
	16 P100	256		

Performance Evaluation

Figure 2 shows the training speed (images/sec) and training time (wall-clock time) of GoogleNet neural network in NV-Caffe using P100 GPUs. It can be seen that the training speed increased as the number of P100 GPUs increased. As a result, the training time decreased. The CPU result in Figure 2 was obtained from Intel-Caffe on two Intel Xeon CPU E5-2690 v4 (14-core Broadwell processors) within one node. We chose Intel-Caffe for the pure CPU test because it has better CPU optimizations than NV-Caffe. From Figure 1, we can see that 1 P100 GPU is **~5.3x** and 4 P100 is **~19.7x** faster than a Broadwell based CPU server. Since NV-Caffe has not supported distributed training so far, we only ran it on up to 4 P100 GPUs on one node.

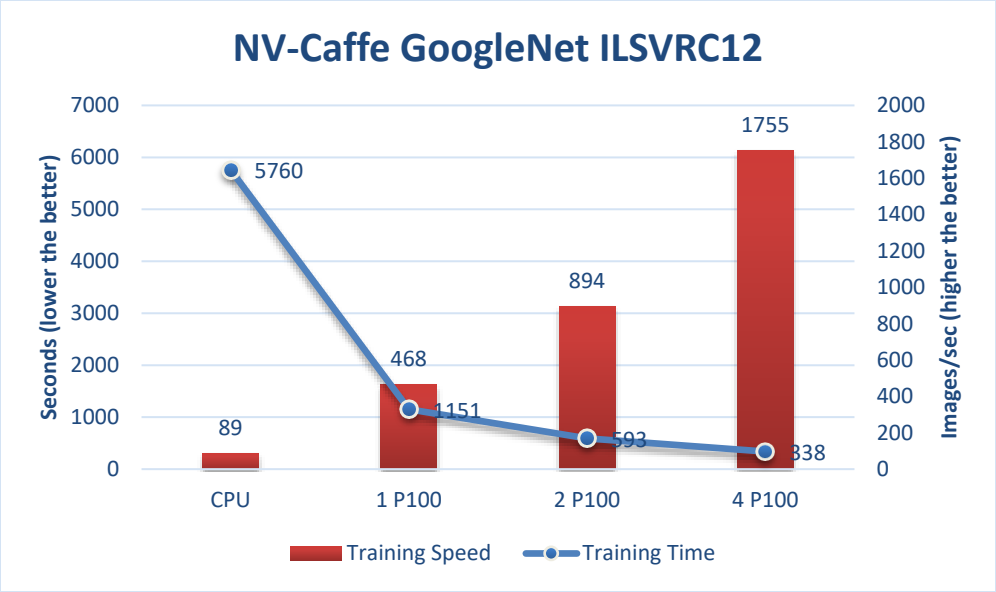

Figure 2: The training speed and time of GoogleNet in NV-Caffe using P100 GPUs

Figure 3 and Figure 4 show the training speed and time of GoogleNet and Inception-BN neural networks in MXNet using P100 GPUs. In both figures, 8 P100 used 2 nodes, 12 P100 used 3 nodes and 16 P100 used 4 nodes. As we can see from both figures, MXNet had great scalability in training speed and training time when more P100 GPUs were used. As mentioned in Section Testing Methodology, if the Ethernet interfaces in all nodes were used, it would impact the training speed and training time significantly since the I/O operation was not fast enough to feed the GPU computations. Based on our observation, the training speed when using Ethernet was only half the speed compared to when using the InfiniBand interfaces. In both MXNet and TensorFlow, the CPU implementation was extremely slow and we believe they were not CPU optimized, therefore we did not compare their P100 performance with CPU performance.

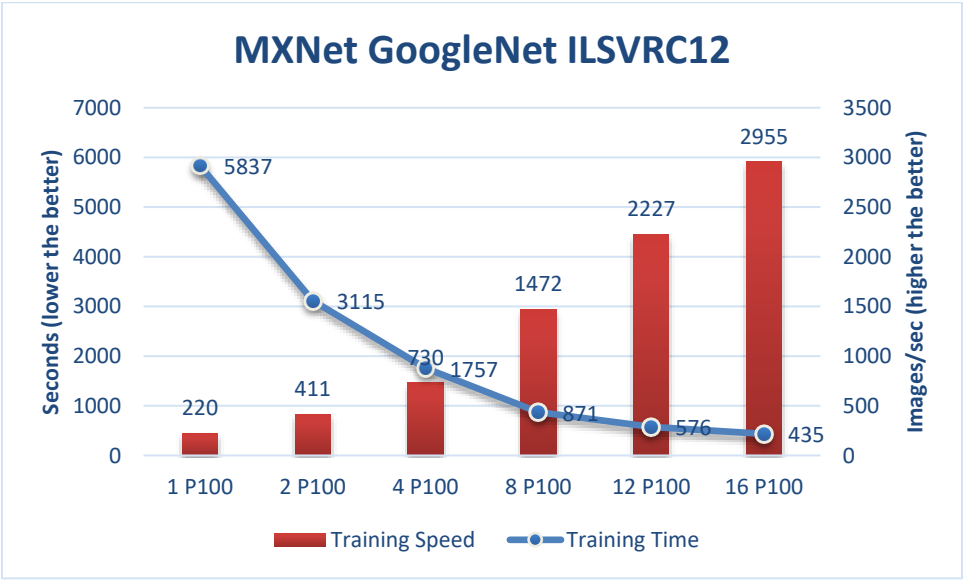

Figure 3: The training speed and time of GoogleNet in MXNet using P100 GPUs

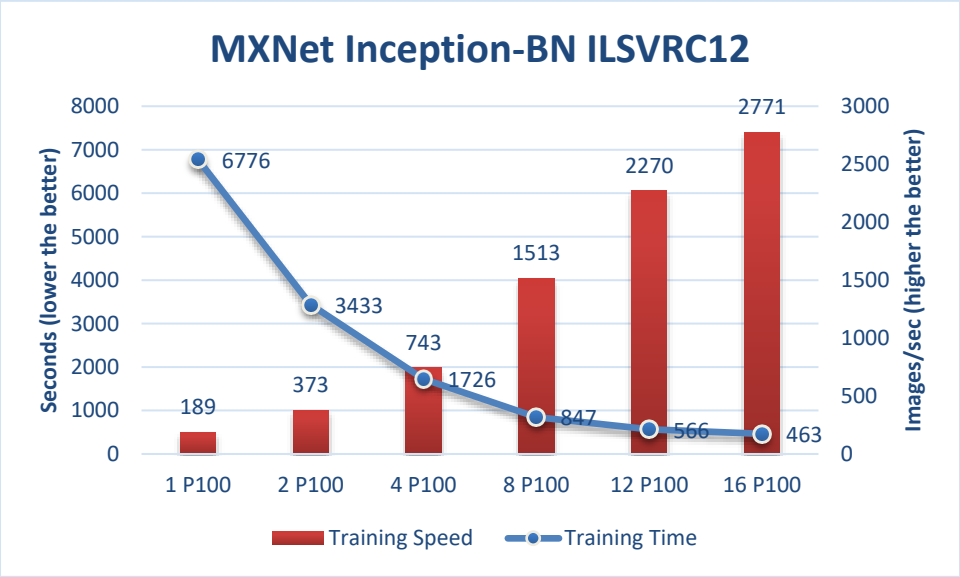

Figure 4: The training speed and time of Inception-BN in MXNet using P100 GPUs

Figure 5 shows the training speed and training time of Inception-V3 neural network in TensorFlow using P100 GPUs. Similar to NV-Caffe and MXNet, TensorFlow also showed good scalability in training speed when more P100 GPUs were used. The training with TensorFlow on multiple nodes was able to run but with poor performance. So that result was not shown here and the reason is still under investigation.

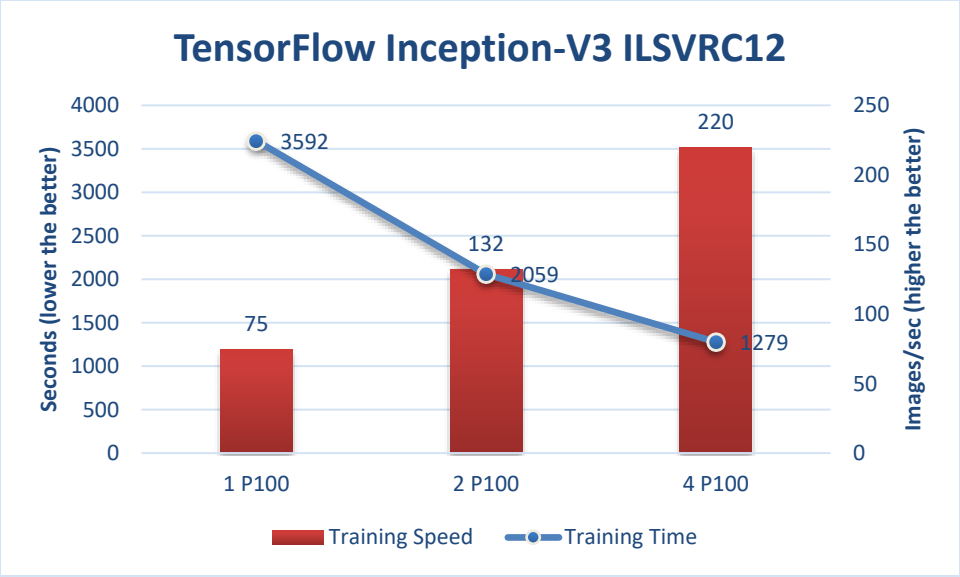

Figure 5: The training speed and time of Inception-V3 in TensorFlow using P100 GPUs

Figure 6 shows the speedup when using multiple P100 GPUs in different deep learning frameworks and neural networks. The purpose of this figure is to demonstrate the speedup in each framework when more number of GPUs are used. The purpose does not include the comparison among different frameworks since their input parameters were different. When using 4 P100 GPUs for NV-Caffe GoogleNet and TensorFlow Inception-V3, we observed a speedup up to 3.8x and 3.0x, respectively. For MXNet, using 16 P100 achieved **13.5x** speedup in GoogleNet and **14.7x** speedup in Inception-BN which are close to the ideal speedup 16x. In particular, we observed linear speedup when using 8 P100 and 12 P100 GPUs in Inception-BN neural network.

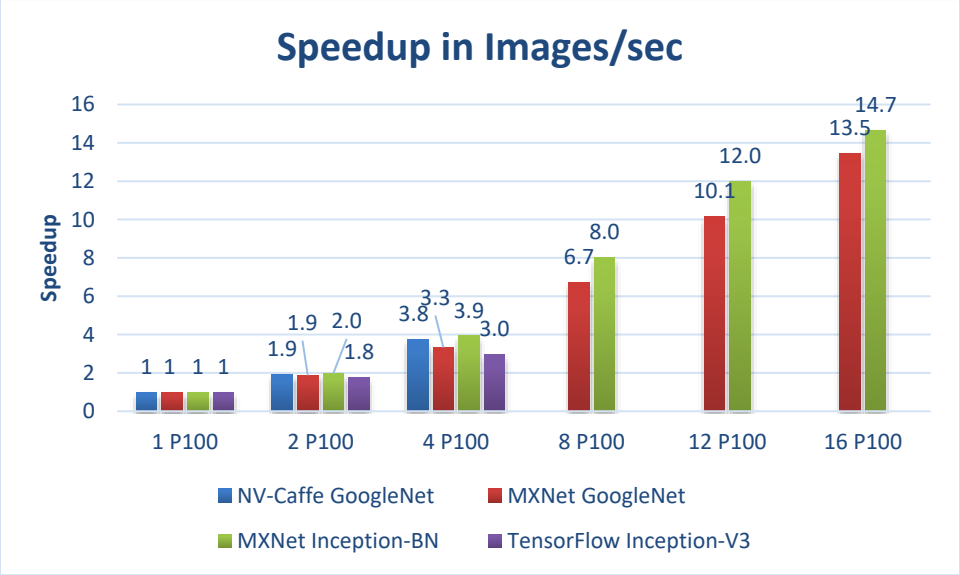

Figure 6: Speedup of multiple P100 GPUs in different DL frameworks and networks

In practice, a real user application can take days or weeks for training a model. Although our benchmarking cases run in a few minutes or a few hours, they are just small snapshots from much longer runs that would be needed to really train a network. For example, the training of a real application might take 90 epochs of 1.2M images. A Dell C4130 with P100 GPUs can turn in results in less than a day, while CPU takes >1 week – that's the real benefits to the end users. The effect for real use case is saving weeks of time per run, not seconds.

Conclusions and Future Work

Overall, we observed great speedup and scalability in neural network training when multiple P100 GPUs were used in Dell's PowerEdge C4130 server and multiple server nodes were used. The training speed increased and the training time decreased as the number of P100 GPUs increased. From the results shown, it is clear that Dell's PowerEdge C4130 cluster is a powerful tool for significantly speeding up neural network training.

In the future work, we will try the P100 for NVLink-optimized servers with the same deep learning frameworks, neural networks and the dataset and see how much performance improvement can be achieved. This blog experimented the PowerEdge C4130 configuration G in which only GPU 1 and GPU 2,

and GPU3 and GPU 4 have peer-to-peer accesses. In the future, we will try C4130 configuration B in which all of the four GPUs connected to one socket have peer-to-peer accesses and check the performance impact in this configuration. We will also investigate the impact of hyperparameters (e.g. batch size and learning rate) on both training performance and model accuracy. The reason of the slow training performance with TensorFlow on multiple nodes will also be examined.