Scaling Deep Learning on Multiple V100 Nodes

Authors: Rengan Xu, Frank Han, Nishanth Dandapanthula.

HPC Innovation Lab. November 2017

Abstract

In our previous blog, we presented the deep learning performance on single Dell PowerEdge C4130 node with four V100 GPUs. For very large neural network models, a single node is still not powerful enough to quickly train those models. Therefore, it is important to scale the training model to multiple nodes to meet the computation demand. In this blog, we will evaluate the multi-node performance of deep learning frameworks MXNet and Caffe2. The results will show that both frameworks scale well on multiple V100-SXM2 nodes.

Overview of MXNet and Caffe2

In this section, we will give an overview about how MXNet and Caffe2 are implemented for distributed training on multiple nodes. Usually there are two ways to parallelize neural network training on multiple devices: data parallelism and model parallelism. In data parallelism, all devices have the same model but different devices work on different pieces of data. While in model parallelism, difference devices have parameters of different layers of a neural network. In this blog, we only focus on the data parallelism in deep learning frameworks and will evaluate the model parallelism in the future. Another choice in most deep learning frameworks is whether to use synchronous or asynchronous weight update. The synchronous implementation aggregates the gradients over all workers in each iteration (or mini-batch) before updating the weights. However, in asynchronous implementation, each worker updates the weight independently with each other. Since the synchronous way guarantees the model convergence while the asynchronous way is still an open question, we only evaluate the synchronous weight update.

MXNet is able to launch jobs on a cluster in several ways including SSH, Yarn, MPI. For this evaluation, SSH was chosen. In SSH mode, the processes in different nodes use rsync to synchronize the working directory from root node into slave nodes. The purpose of synchronization is to aggregate the gradients over all workers in each iteration (or mini-batch). Caffe2 uses Gloo library for multi-node training and Redis library to facilitate management of nodes in distributed training. Gloo is a MPI like library that comes with a number of collective operations like barrier, broadcast and allreduce for machine learning applications. The Redis library used by Gloo is used to connect all participating nodes.

Testing Methodology

We chose to evaluate two deep learning frameworks for our testing, MXNet and Caffe2. As with our previous benchmarks, we will again use the ILSVRC 2012 dataset which contains 1,281,167 training images and 50,000 validation images. The neural network in the training is called Resnet50 which is a computationally intensive network that both frameworks support. To get the best performance, CUDA 9 compiler, CUDNN 7 library and NCCL 2.0 are used for both frameworks, since they are optimized for V100 GPUs. The testing platform has four Dell EMC's PowerEdge C4130 servers in configuration K. The system layout of configuration K is shown in Figure 1. As we can see, the server has four V100-SXM2 GPUs and all GPUs are connected by NVLink. The other hardware and software details are shown in Table 1. Table 2 shows the input parameters that are used to train Resnet50 neural network in both frameworks.

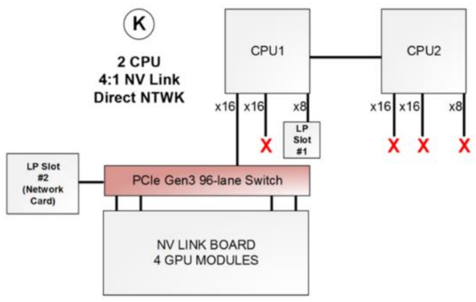

Figure 1: C4130 configuration K

Table 1: The hardware configuration and software details

Platform	**PowerEdge C4130 (configuration K)**
CPU	2 x Intel Xeon E5-2690 v4 @2.6GHz (Broadwell)
Memory	256GB DDR4 @ 2400MHz
Disk	9TB HDD
GPU	V100-SXM2
Software and Firmware	
Operating System	RHEL 7.3 x86_64
Linux Kernel	3.10.0-514.26.2.el7.x86_64
BIOS	2.4.2
CUDA compiler and GPU driver	CUDA 9.0 (384.81)
Python	2.7.5
Interconnect	
OFED	**4.1-1.0.2**
Deep Learning Libraries and Frameworks	
CUDNN	7.0
NCCL	2.0
MXNet	0.11.1
Caffe2	0.8.1+

Table 2: Input parameters used in different deep learning frameworks

		Batch Size	**Epochs**
MXNet	FP32	64 per GPU	1 epoch
	FP16	128 per GPU	1 epoch
Caffe2	FP32	64 per GPU	1 epoch
	FP16	128 per GPU	1 epoch

Performance Evaluation

Figure 2 and Figure 3 show the Resnet50 performance and speedup results on multiple nodes with MXNet and Caffe2, respectively. As we can see, the performance scales very well with both frameworks. With MXNet, compared to 1*V100, the speedup of using 16*V100 (in 4 nodes) is **15.4x** in FP32 mode and **13.8x** in FP16, respectively. And compared to FP32, FP16 improved the performance by 63.28% - 82.79%. Such performance improvement was contributed to the Tensor Cores in V100.

In Caffe2, compared to 1*V100, the speedup of using 16*V100 (4 nodes) is **14.8x** in FP32 and **13.6x** in FP16, respectively. And the performance improvement of using FP16 compared to FP32 is 50.42% - 63.75% excluding the 12*V100 case. With 12*V100, using FP16 is only 29.26% faster than using FP32. We are still investigating the exact reason of it, but one possible explanation is that 12 is not the power of 2, which may make some operations like reductions slower.

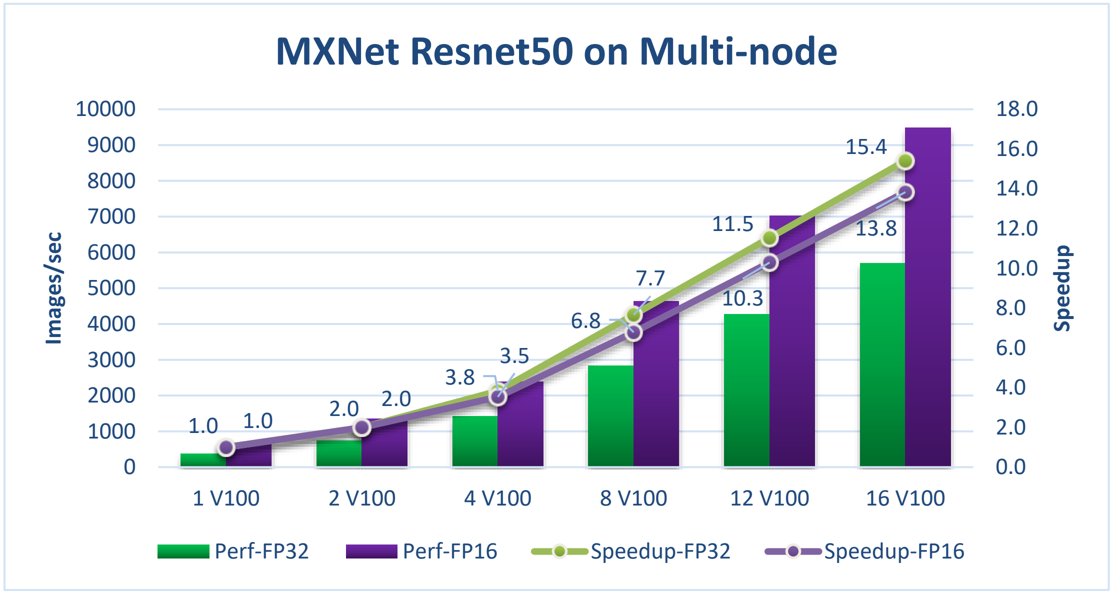

Figure 2: Performance of MXNet Resnet50 on multiple nodes

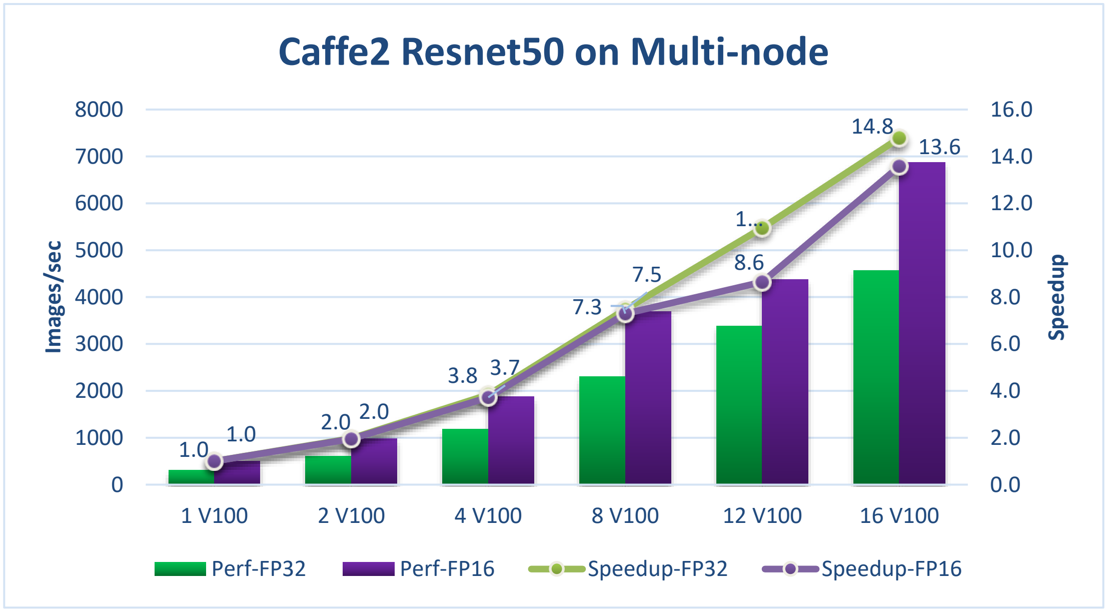

Figure 3: Performance of Caffe2 Resnet50 on multiple nodes

Conclusions and Future Work

In this blog, we present the performance of MXNet and Caffe2 on multiple V100-SXM2 nodes. The results demonstrate that the deep learning frameworks are able to scale very well on multiple Dell EMC's PowerEdge servers. At this time the FP16 support in TensorFlow is still experimental, our evaluation is in progress and the results will be included in future blogs. We are also working on containerizing these frameworks with Singularity to make their deployment much easier.