Deep Learning on V100

Authors: Rengan Xu, Frank Han, Nishanth Dandapanthula.

HPC Innovation Lab. September 2017

Overview

In this blog, we will introduce the NVIDIA Tesla Volta-based V100 GPU and evaluate it with different deep learning frameworks. We will compare the performance of the V100 and P100 GPUs. We will also evaluate two types of V100: V100-PCIe and V100-SXM2. The results indicate that in training V100 is ~40% faster than P100 with FP32 and >100% faster than P100 with FP16, and in inference V100 is 3.7x faster than P100. This is one blog of our Tesla V100 blog series. Another blog of this series is about the general HPC applications performance on V100 and you can read it here.

Introduction to V100 GPU

In the 2017 GPU Technology Conference (GTC), NVIDIA announced the Volta-based V100 GPU. Similar to P100, there are also two types of V100: V100-PCIe and V100-SXM2. V100-PCIe GPUs are inter-connected by PCIe buses and the bi-directional bandwidth is up to 32 GB/s. V100-SXM2 GPUs are inter-connected by NVLink and each GPU has six links and the bi-directional bandwidth of each link is 50 GB/s, so the bi-directional bandwidth between different GPUs is up to 300 GB/s. A new type of core added in V100 is called tensor core which was designed specifically for deep learning. These cores are essentially a collection of ALUs for performing 4x4 matrix operations: specifically a fused multiply add (A*B+C), multiplying two 4x4 FP16 matrices together, and then adding to a FP16/FP32 4x4 matrix to generate a final 4x4 FP16/FP32 matrix. By fusing matrix multiplication and add in one unit, the GPU can achieve high FLOPS for this operation. A single Tensor Core performs the equivalent of 64 FMA operations per clock (for 128 FLOPS total), and with 8 such cores per Streaming Multiprocessor (SM), 1024 FLOPS per clock per SM. By comparison, even with pure FP16 operations, the standard CUDA cores in a SM only generate 256 FLOPS per clock. So in scenarios where these cores can be used, V100 is able to deliver 4x the performance versus P100. The detailed comparison between V100 and P100 is in Table 1.

Table 1: The comparison between V100 and P100

	P100-PCIe	**V100-PCIe**	**Improvement**	**P100-SXM2**	**V100-SXM2**	**Improvement**
CUDA Cores	3584	5120		3584	5120	
Tensor Cores	N/A	640		N/A	640	
Boost Clock	1329 MHz	1380 MHz		1481 MHz	1530 MHz	
Memory Bandwidth	732 GB/s	900 GB/s	22.95%	732 GB/s	900 GB/s	22.95%
NVLink Bi-bandwidth	N/A	N/A		160 GB/s	300 GB/s	
Double Precision	4.7 TFLOPS	7 TFLOPS	1.5x	5.3 TFLOPS	7.8 TFLOPS	1.5x
Single Precision	9.3 TFLOPS	14 TFLOPS	1.5x	10.6 TFLOPS	15.7 TFLOPS	1.5x
Deep Learning	18.6 TFLOPS	112 TFLOPS	6x	21.2 TFLOPS	125 TFLOPS	6x
Architecture	Pascal	Volta		Pascal	Volta	
TDP	250W	250W		300W	300W	

Testing Methodology

As in our previous deep learning blog, we still use the three most popular deep learning frameworks: NVIDIA's fork of Caffe (NV-Caffe), MXNet and TensorFlow. Both NV-Caffe and MXNet have been optimized for V100. TensorFlow still does not have any official release to support V100, but we applied some patches obtained from TensorFlow developers so that it is also optimized for V100 in these tests. For the dataset, we still use ILSVRC 2012 dataset whose training set contains 1281167 training images and 50000 validation images. When testing neural network, we chose Resnet50 as it is a computationally intensive network. To get best performance, we used CUDA 9-rc compiler and CUDNN library in all of the three frameworks since they are optimized for V100. The testing platform is Dell EMC's PowerEdge C4130 server. The C4130 server has multiple configurations, and we evaluated both P100-PCIe in configuration G and P100-SXM2 in configuration K. The difference between configuration G and configuration K is shown in Figure 1. There are mainly two differences: one is that configuration G has two x16 PCIe links connecting from dual CPUs to the four GPUs, while configuration K has only one x16 PCIe bus connecting from one CPU to four GPUs; another difference is that GPUs are connected by PCIe buses in configuration G but by NVLink in configuration K. The other hardware and software details are shown in Table 2.

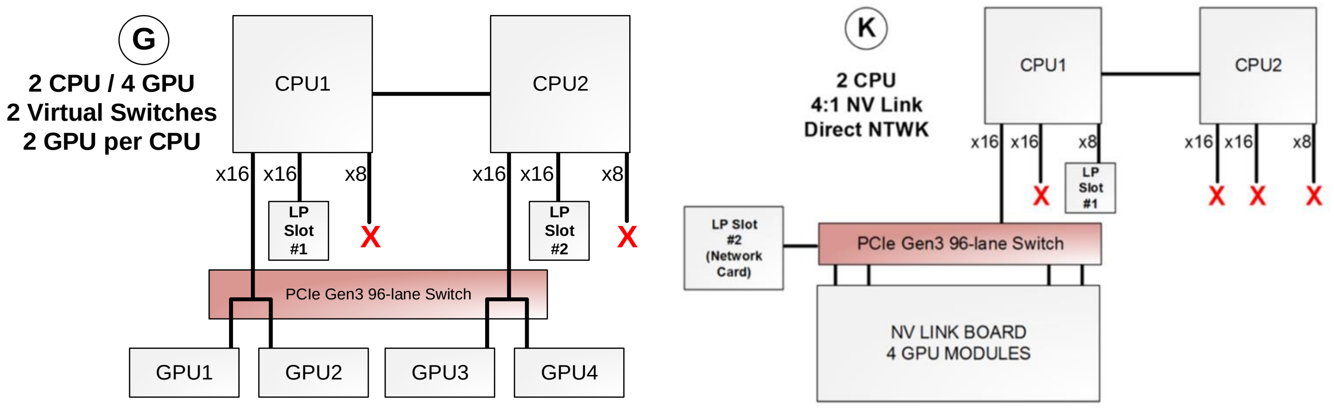

Figure 1: Comparison between configure G and configure K

Table 2: The hardware configuration and software details

Platform	**PowerEdge C4130 (config G and config K)**
CPU	2 x Intel Xeon E5-2690 v4 @2.6GHz (Broadwell)
Memory	256GB DDR4 @ 2400MHz
Disk	9TB HDD
GPU	V100-PCIe, V100-SXM2, P100-PCIe, P100-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-RC (384.59)
NCCL	2.0
Python	2.7.5
Deep Learning Libraries and Frameworks	
CUDNN	7.0
TensorRT	3.0.0
NV-Caffe	0.16.3
MXNet	0.11.0
TensorFlow	1.2.1-rc1

In this experiment, we trained various deep learning frameworks with one pass on the whole dataset since we were comparing only the training speed, not the training accuracy. Other important input parameters for different deep learning frameworks are listed in Table 3. For NV-Caffe and MXNet, in terms of different batch size, we doubled the batch size for FP16 tests since FP16 consumes half the memory for floating points as FP32. As TensorFlow does not support FP16 yet, we did not evaluate its FP16 performance in this blog. Because of different implementations, NV-Caffe consumes more memory than MXNet and TensorFlow for the same neural network, the batch size in FP32 mode is only half of that in MXNet and TensorFlow. In NV-Caffe, if FP16 is used, then the data type of several parameters need to be changed. We explain these parameters as follows: the solver_data_type controls the data type for master weights; the default_forward_type and default_backward_type controls the data type for training values; the default_forward_math and default_backward_math controls the data type for matrix-multiply accumulator. In this blog we used FP16 for training values, FP32 for matrix-multiply accumulator and FP32 for master weights. We will explore other combinations in our future blogs. In MXNet, we tried different values for the parameter "--data-nthreads" which controls the number of threads for data decoding.

Table 3: Input parameters used in different deep learning frameworks

		Batch Size	Iterations/Epochs
NV-Caffe	FP32	128	10000 iterations
	FP16	256	5000 iterations
MXNet	FP32	256	5000 iterations
	FP16	512	2500 iterations
TensorFlow	FP32	256	5000 iterations

Performance Evaluation

Figure 1, Figure 2, and Figure 3 show the performance of V100 versus P100 with NV-Caffe, MXNet and TensorFlow, respectively. And Table 4 shows the performance improvement of V100 compared to P100. From these results, we can obtain the following conclusions:

- In both PCIe and SXM2 versions, V100 is >40% faster than P100 in FP32 for both NV-Caffe and MXNet. This matches the theoretical speedup. Because FP32 is single precision floating points, and V100 is 1.5x faster than P100 in single precision. With TensorFlow, V100 is more than 30% faster than P100. Its performance improvement is lower than the other two frameworks and we think that is because of different algorithm implementations in these frameworks.

- In both PCIe and SXM2 versions, V100 is >2x faster than P100 in FP16. Based on the specification, V100 tensor performance is ~6x than P100 FP16. The reason that the actual speedup does not match the theoretical speedup is that not all data are stored in FP16 and so not all operations are tensor operations (the FMA matrix multiply and add operation).

- In V100, the performance of FP16 is close to 2x that of FP32. This is because FP16 only requires half storage compared to FP32 and therefore we could double the batch size in FP16 to improve the computation speed.

- In MXNet, we set the "--data-nthreads" to 16 instead of the default value 4. The default value is often sufficient to decode more than 1K images per second but still not fast enough for V100 GPU. In our testing, we found the default value 4 is enough for P100 but for V100 we need to set it at least 12 to achieve good performance, with a value of 16 being ideal.

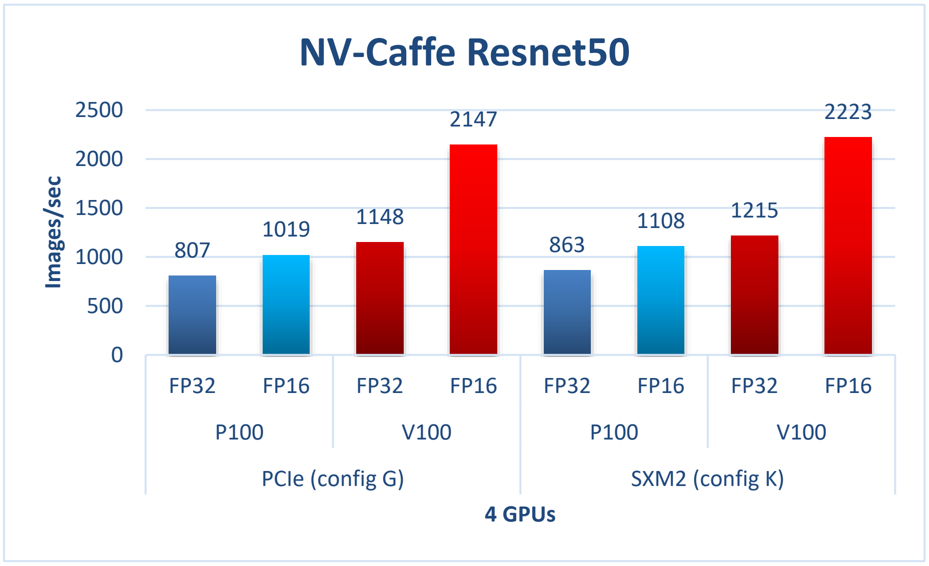

Figure 2: Performance of V100 vs P100 with NV-Caffe

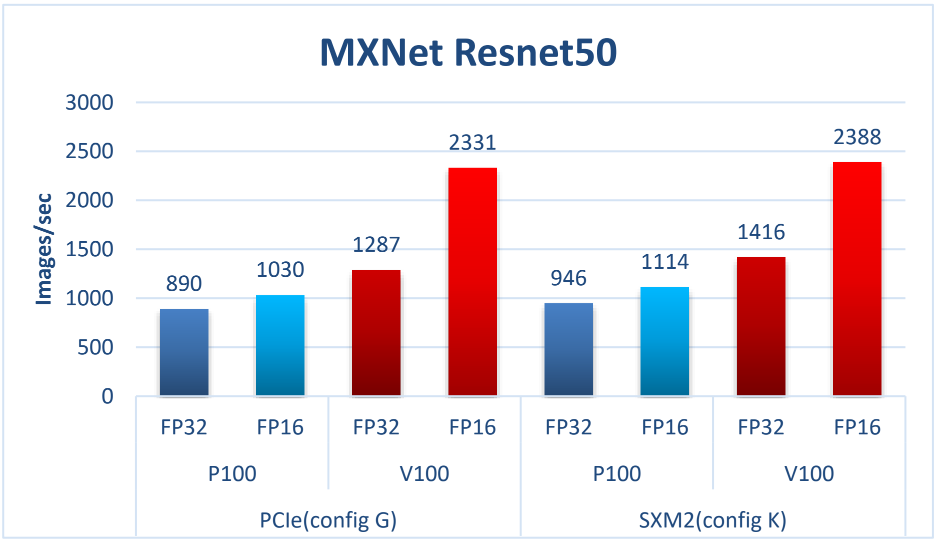

Figure 3: Performance of V100 vs P100 with MXNet

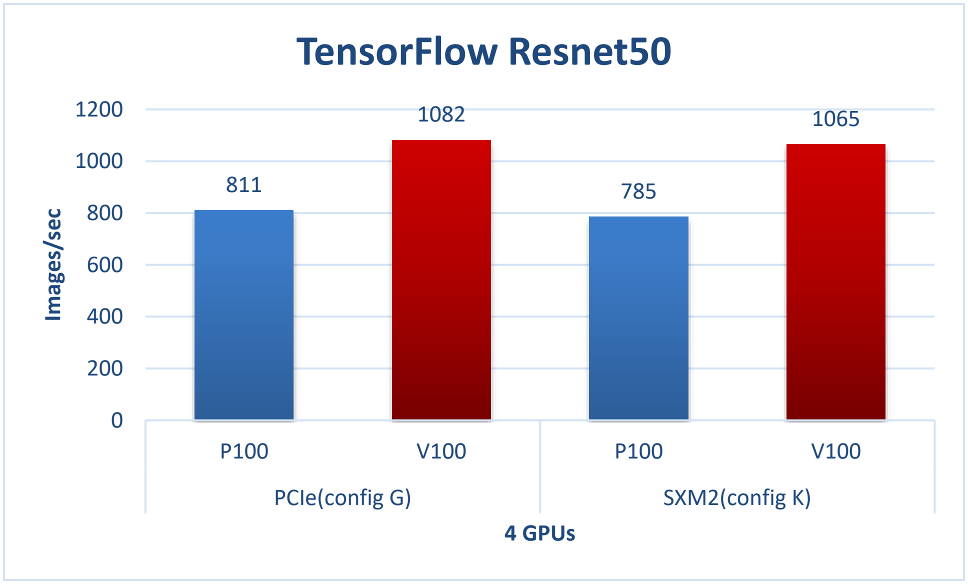

Figure 4: Performance of V100 vs P100 with TensorFlow

Table 4: Improvement of V100 compared to P100

		Batch Size	V100 vs P100 (%)
PCIe	FP32	NV-Caffe	42.23%
		MXNet	44.67%
		TensorFlow	31.60%
	FP16	NV-Caffe	110.68%
		MXNet	126.26%
SXM2	FP32	NV-Caffe	40.72%
		MXNet	49.76%
		TensorFlow	33.14%
	FP16	NV-Caffe	100.56%
		MXNet	114.33%

Since V100 supports both deep learning training and inference, we also tested the inference performance with V100 using the latest TensorRT 3.0.0. The testing was done in FP16 mode on both V100-SXM2 and P100-PCIe and the result is shown in Figure 5. We used batch size 39 for V100 and 10 for P100. Different batches were chosen to make their inference latencies are close to each other (~7ms in the figure). The result shows that when their latencies are close, the inference throughput of V100 is 3.7x faster compared to P100.

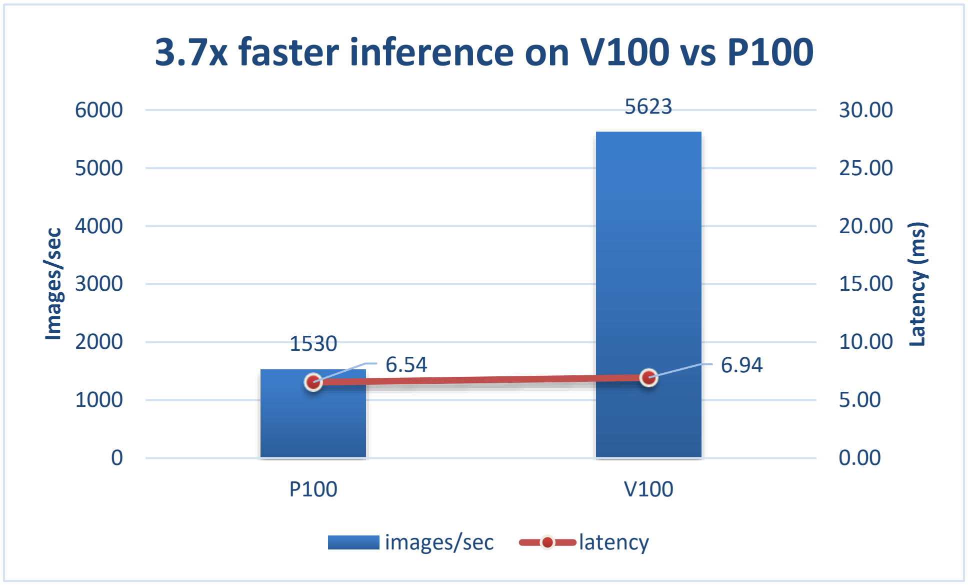

Figure 5: Resnet50 inference performance on V100 vs P100.

Conclusions and Future Work

After evaluating the performance of V100 with three popular deep learning frameworks, we conclude that in training V100 is more than 40% faster than P100 in FP32 and more than 100% faster in FP16, and in inference V100 is 3.7x faster than P100. This demonstrates the performance benefits when the V100 tensor cores are used. In the future work, we will evaluate different data type combinations in FP16 and study the accuracy impact with FP16 in deep learning training. We will also evaluate the TensorFlow with FP16 once support is added into the software. Finally, we plan to scale the training to multiple nodes with these frameworks.