Deep Learning at Scale on NVIDIA V100 Accelerators

Rengan Xu, Frank Han and Quy Ta
AI Engineering, Server and Infrastructure Systems
Dell EMC
Austin, TX, United States
{Rengan.Xu, Frank.Han, Quy.Ta}@Dell.com

***Abstract*—The recent explosion in the popularity of Deep Learning (DL) is due to a combination of improved algorithms, access to large datasets and increased computational power. This had led to a plethora of open-source DL frameworks, each with varying characteristics and capabilities. End users are then left with the difficult task of determining software and hardware configurations to get optimal performance from each framework.**

We share our experiences and develop best practices for DL training with TensorFlow, MXNet and Caffe2. The paper also looks at DL inferencing with TensorRT on NVIDIA V100 Volta GPUs. It focuses on one of the more prominent neural network architectures, Resnet50, combined with Imagenet dataset. We quantify the impact of hardware attributes on DL workloads such as the usage of PCIe vs NVLink GPUs, performance past a single worker node, effect of high speed interconnect such as InfiniBand EDR on training and the implication of utilizing a network attached storage and its advantages.

***Index Terms*—Deep Learning, Distributed Training, GPU, Benchmarking, V100**

I. Introduction

In recent years, Deep Learning (DL) [1] has achieved great success in many fields such as computer vision, speech recognition, natural language processing and so on. The recent popularity of deep learning is due to a combination of improved algorithms, access to large data sets and increased computational power. The popularity of DL has led to a plethora of open-source DL frameworks such as TensorFlow [2] from Google, Apache MXNet [3], and Caffe2 [4] from Facebook. All of these frameworks can utilize GPUs to accelerate the compute intensive portions of the neural network models. Traditional deep learning models targeted at image related use cases comprise of a lot of matrix multiply operations and the GPU architecture is ideal to parallelize these operations and therefore reducing the model training time significantly.

Since each DL framework has its own characteristics, the end users are then left with the difficult task of determining the software and hardware configurations to get the optimal performance from each framework. In this paper, we chose a computationally intensive model and train it under a variety of conditions and frameworks in order to draw insight and develop a set of best practices for each. We used TensorFlow, MXNet and Caffe2. All tests were run on a Dell EMC cluster containing multiple nodes with Nvidia V100 "Volta" GPUs. We will investigate whether they scale well on the cluster and tune the runtime parameters to ensure they scale as best as possible. If there are still issues that prevent the scaling, then we profile the application and analyze the possible reasons for this behavior. Profiling allows us to ascertain the CPU, memory, network and storage demands for that particular neural network model and framework.

Deep Learning consists of two main phases: training and inference. Besides the training experiments we just mentioned, inference also plays a key role as it is the ultimate goal of any deep learning exercise. For inference, we will use Nvidia's TensorRT library as it not only supports inference with the commonly used 32-bit floating point (FP32), but also with 8-bit integers (INT8). We will compare the throughput advantage of INT8, and highlight the negligible difference in accuracy when compared to FP32.

The main contributions of this paper include:

- We benchmarked deep learning training on V100 GPUs at scale (past a single worker node) using different frameworks and libraries and provided a comprehensive comparison and analysis. This relates to the usage of the CPU, memory, network and storage.
- The performance difference for DL workloads of V100 GPUs using NVLink and PCIe GPU interconnects are compared.
- The training performance with FP32 and the reduced precision FP16 are compared.
- The characteristics of storage and node interconnect have been measured and analyzed.
- We compared the deep learning inference throughput with FP32 and INT8, and conclude that INT8 can also achieve the comparable accuracy with FP32.

The organization of this paper is as follows: Section II discusses the related work. Section III introduces V100 GPUs and several deep learning frameworks. Section IV provides different types of tuning and analysis on DL training with TensorFlow, MXNet and Caffe2 frameworks. Then Section V presents the inference throughput and accuracy comparison between INT8 and FP32. We conclude our work in Section VI.

II. Related Work

Shi et al. [5] provided a comprehensive comparison for different deep learning frameworks including Caffe, CNTK [6], TensorFlow, and Torch [7]. They benchmarked those frameworks on CPU, single GPU and multi-GPUs within a single

node. Their GPU tests used the Pascal GPU. In contrast we perform multi-GPU tests on single node, and scale up to eight nodes and 32 GPUs. We also use the latest Volta GPUs which include many features geared specifically for DL applications (will be discussed in Section III). In addition, we profile the frameworks to get an idea of the particular resource utilization. DAWNBench [8] proposed a benchmark that focused on end-to-end training time to achieve a state-of-the-art accuracy, as well as maintaining the accuracy in the inference phase. The paper's goal was to provide a means of evaluating many trade offs in deep learning systems. Uber's Horovod [9] first evaluated the TensorFlow's own distributed implementation and found that there is a large gap between the actual speedup and the ideal speedup. Then they went on to implement an MPI overlay for TensorFlow and achieved 88% efficiency. These experiments were performed in a P100 GPU cluster where all nodes are connected with 25Gbps RDMA-capable Ethernet. Because of its high scaling efficiency, we leverage Horovod framework for running TensorFlow across multiple nodes, but our experiments are on a V100 GPU cluster where all nodes are connected with 100Gbps RDMA-capable InfiniBand network. Goyal et al. [10] applied several techniques to train Resnet-50 model with a minibatch size of 8192 on 256 GPUs, while still matching small minibatch accuracy. You et al. [11] took only 14 minutes to train Resnet-50 to achieve 74.9% top-1 accuracy with 512 Intel Knights Landing coprocessors.

III. Background

During the 2017 GPU Technology Conference (GTC), NVIDIA announced the Volta-based V100 GPU, of which there are two types: 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 with the bi-directional bandwidth of each link being 50 GB/s. The bi-directional bandwidth for the NVLink implementation between GPUs is up to 300 GB/s which is 9.4 times more than the bandwidth available in the PCIe implementation. The V100 is the first Nvidia GPU to contain "Tensor Cores" which are cores specifically designed for 4 x 4 matrix multiplication operations which are a major part of DL models. These cores are essentially a collection of ALUs for performing 4x4 matrix operations: specifically a fused multiply add (FMA). To elaborate, this translates to three matrices A, B and C and the operation being 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). With 8 such cores per Streaming Multiprocessor (SM), 1024 FLOPS per clock per SM can be achieved.

The table I looks at the specification differences between the PCIe and SXM2 V100 GPUs. The major difference includes the difference in frequency and the connectivity between GPUs.

TABLE I: V100-PCIe vs V100-SXM2

Description	V100-PCIe	V100-SXM2
CUDA Cores	5120	5120
GPU Max Clock Rate (HMz)	1380	1530
Tensor Cores	640	640
Memory Bandwidth (GB/s)	900	900
NVLink bandwidth (GB/s)	N/A	300
Deep Learning (Tensor OPS)	112	120

TensorFlow, developed by Googles Brain team, is a library for numerical computation using data flow graphs. TensorFlow supports multiple GPUs and can scale to multiple nodes using gRPC [12]. However, we didn't use TensorFlow's distributed implementation as it does not scale well as was discovered in paper [9]. Instead we use Uber's Horovod framework which uses MPI to distribute the computation across multiple worker 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. MXNet is capable of launching jobs on a cluster in several ways including: SSH, Yarn, and MPI. For this evaluation, SSH was chosen. In SSH mode, the processes in different nodes use rsync to synchronize the working directory between the root and worker nodes.

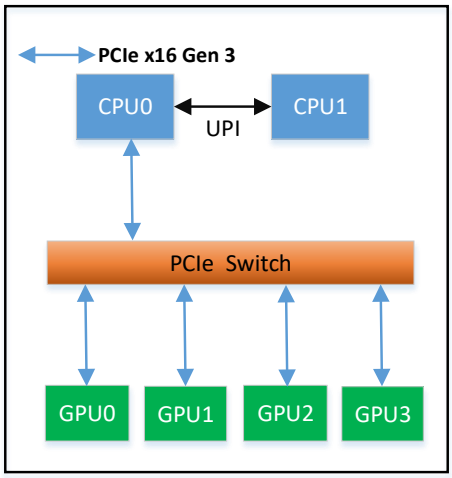
Caffe2, developed by Facebook, is the successor of Caffe. It redesigned the data structures and added support for distributed training on multiple nodes. It provides capabilities to run models on mobile devices. Caffe2 uses the Gloo library [13] for multi-node training and Redis [14] to facilitate management of nodes in distributed training. Gloo is an MPI like library that comes with a number of collective operations like barrier, broadcast, and allreduce for machine learning applications. Redis is used by Gloo to connect all the participating nodes. Recently Caffe2 has been merged with PyTorch [15], but the measurement and analysis approachs presented in this paper are applicable to any framework, regardless whether that framework will disappear in the future or not.

IV. Performance Tuning for DL Training

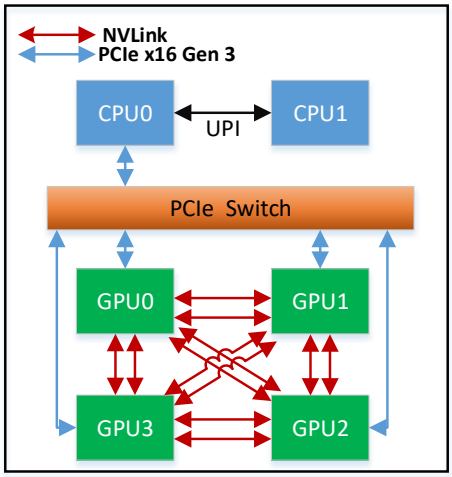
In this section, we tune the deep learning training performance for TensorFlow, MXNet and Caffe2. The experiments were done on a Dell EMC PowerEdge C4140 cluster. The cluster has 10 C4140 nodes which are connected by 100 Gbps InfiniBand EDR network. Each node has dual Intel Xeon Gold 6148 CPUs (Skylake architecture) and four Nvidia V100 GPUs. Of the 10 C4140 nodes, 8 of them have V100-PCIe GPUs and the remaining two have V100-SXM2 GPUs. Figure 1 shows a block diagram of the connections between CPUs and GPUs for both configurations. In each configuration, there is one PCIe bus connecting from one CPU to all four GPUs and therefore the GPUs are in the same PCIe root complex. As a result, all the GPUs are able to do Peer-to-Peer (P2P) memory accesses. The difference in these two configurations is that the V100-PCIe GPUs within a node are

TABLE II: Hardware and software configuration

Platform	PowerEdge C4140
CPU	2x Intel Xeon Gold 6148 (Skylake)
Memory	192GB DDR4 @ 26667MHz
Storage	9TB NFS through IPoIB on EDR InfiniBand
GPU	V100-PCIe, V100-SXM2, with 16GB memory
Node Interconnects	EDR 100 Gbps InfiniBand
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 driver	CUDA 9.0 (387.26)
Deep Learning Frameworks and Libraries	
CUDNN	7.0
NCCL	2.0
TensorRT	4.0.0
Horovod	0.11.3
TensorFlow	1.5
MXNet	0.11.1
Caffe2	0.8.1



(a) V100-PCIe



(b) V100-SXM2

Fig. 1: C4140 server with V100 GPUs

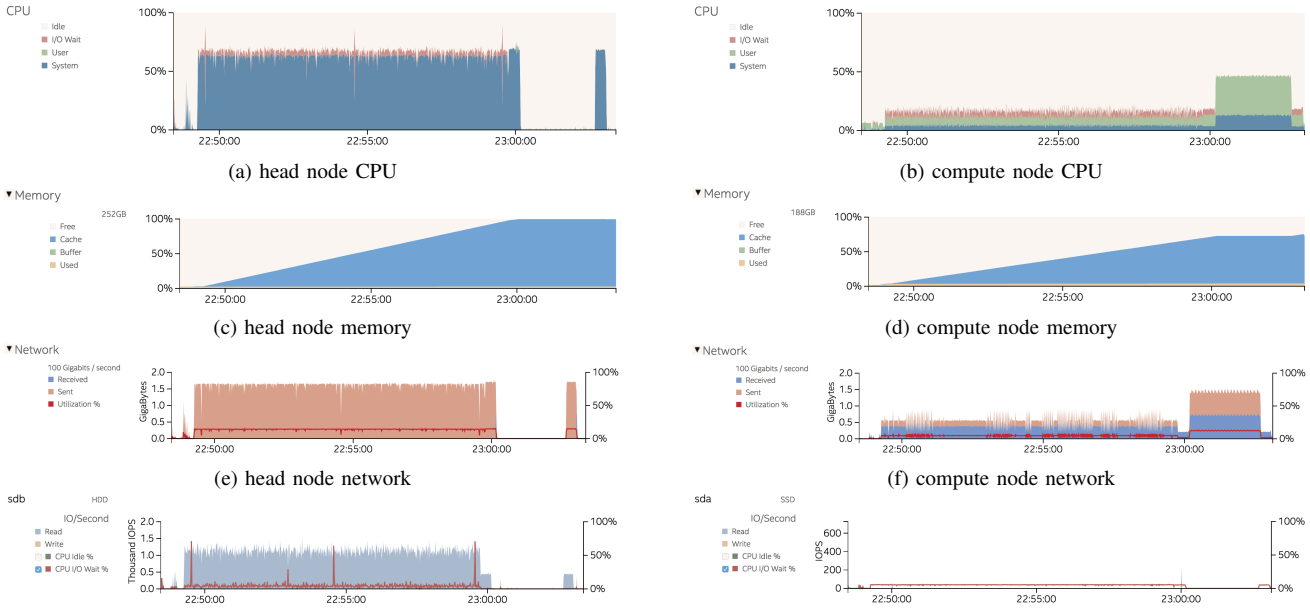
connected to each other through PCIe, while the V100-SXM2 GPUs are connected through NVLink. The storage server is a Dell PowerVault MD3220 with 9TB of usable disk. It is connected to the head node of the cluster via SAS HBA. All the compute nodes mount this storage using InfiniBand EDR. The hardware and software details of the testbed are shown in Table II.

We use the well known ILSVRC 2012 as our benchmark data set [16]. This data set contains 1,281,167 training images and 50,000 validation images. All images are grouped into 1,000 categories or classes. The total size of the whole dataset is 143GB. The overall size of ILSVRC 2012 leads to non-trivial training times and makes it more interesting for analysis. Additionally, the data set is commonly used by DL researchers for benchmarking and can provide a comparative view point. For the neural network model, we chose Resnet50 [17] as it is a computationally intensive network. For the batch size parameter in deep learning, we chose the maximum batch size that does not cause GPU memory errors. In our testing, the batch size is 64 per GPU for MXNet and Caffe2, and 128 per GPU for TensorFlow. We measured the performance in images/sec which is a measure of throughput. The images/sec number was averaged across all iterations to take into account the minute run to run deviations. The training tests were run for a single epoch, or one pass through the entire data set. The throughput is stable through epochs. However, this does not apply to Caffe2 and we will explain this result in Section IV-A. Training tests were run on different node counts from a single node to eight nodes, while inference was only run on a single node. We also measure the actual network bandwidth for the InfiniBand fabric bandwidth and disk throughput when running the Resnet50 model.

A. *Single GPU vs Multi-GPU*

Figure 2 shows the scaling performance and speedup of one to 32 V100-PCIe GPUs for all three frameworks. The multiple GPUs are either in single node (up to 4 GPUs) or across eight nodes. It can be seen that among the three frameworks, with 32 GPUs, MXNet scales the best with 29.43x in FP32 mode and 25.84x in FP16 mode at a 32 GPU scale relative to the performance of a single GPU. TensorFlow also scales well with 21.95x in FP32 mode and 23.72x in FP16 mode. Caffe2 scales well within a single node for which the speedup is 3.55x in FP32 and 3.58x in FP16 mode at a four GPU scale relative to a single GPU. However, it does not scale well beyond a single node. This result is with Redis library for multi-node communication. The performance varies between 200 images/sec to 1,000 images/sec in each iteration. We have also tried its MPI implementation and the result is similar to the Redis implementation.

To find out why Caffe2 does not perform as expected, we



(a) head node CPU

(b) compute node CPU

(c) head node memory

(d) compute node memory

(e) head node network

(f) compute node network

(g) head node disk

(h) compute node disk

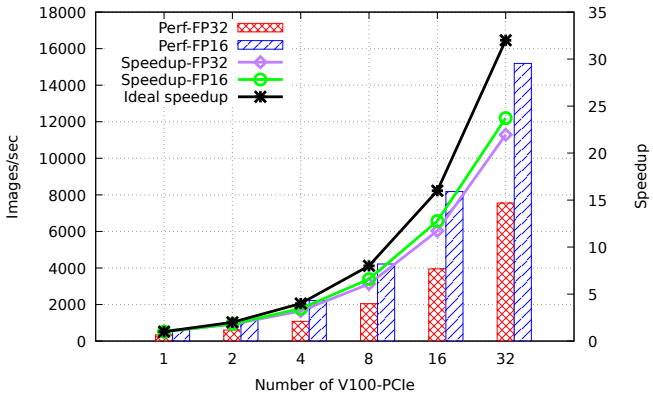
Fig. 3: Caffe2 performance profiling on both head node and compute nodes of a cluster

profiled the GPU portion of the application with nvprof [18] profiler and the CPU portion with Intel Storage Performance Snapshot (SPS) [19]. We found the issue is on the CPU side. We profiled two training epochs on both the head node and all compute nodes and the profiling result is shown in Figure 3. The profiling on all compute nodes are similar and Figure 3 only shows one compute node as an example. It is clear that the system caches the data in memory when the program is reading the image database from the disk. In compute nodes, the utilization of both CPU and InfiniBand network is low when the database is not fully cached. During this stage, because of the I/O wait on CPU, the training speed is unstable. But after the database is fully cached, the utilization of both CPU and InfiniBand network is much higher as both of them do not need to wait for the data reading from the disk, instead the data is read from memory. Since the compute nodes read the data from network file system (NFS), as shown in Figure 3h the local disk does not have any reading/writing operations. Based on this profiling data, we conclude that the training speed on multiple nodes is as good as the training speed on one node. In our experiment, the size of ILSVRC2012 image database for Caffe2 is $\sim$ 260GB which is lower than the head node and compute node memory size. If the nodes cannot cache the whole image database, then the performance will be as poor as the performance in the first epoch. More storage experiments will be presented in Section IV-E.

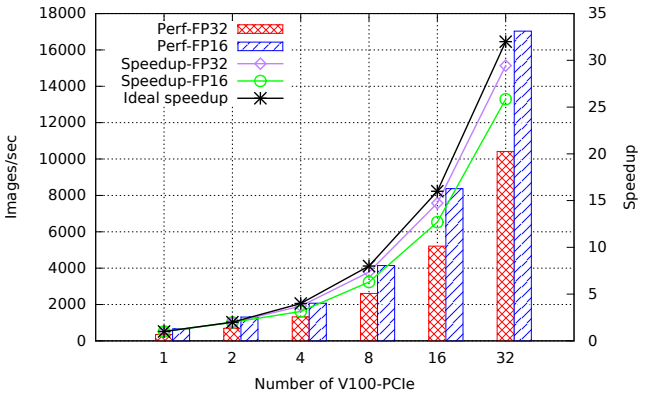
B. V100-SXM2 vs V100-PCIe

As previously shown in Table I, the main difference between SXM2 and PCIe is that the SXM2 has higher clock frequency and higher bandwidth between GPUs. To demonstrate how much improvement can be gained from V100-SXM2 compared to V100-PCIe, we ran the three frameworks on these two different types of GPUs using the same dataset and parameter settings. Figure 4 shows the performance differences between these two GPU types. The comparison is up to 8 GPUs since we have only 2 nodes with V100-SXM2. When a single GPU is used, the only difference between SXM2 and PCIe version is the clock frequency. The GPU max clock rate is 1,380 MHz for PCIe and 1,530 MHz for SXM2 which is $\sim$10.9% higher than PCIe. The performance improvement among three frameworks with SXM2 for single GPU is 3.6%-6.9% which is in the reasonable range.

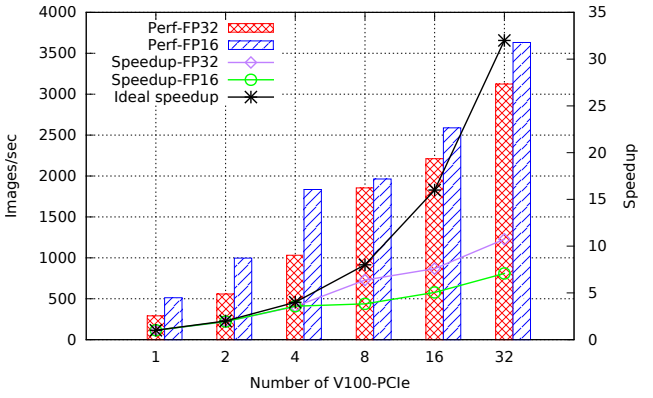
When multi-GPU is used, V100-SXM2 has the advantage of using NVLink over the V100-PCIe which uses PCIe buses for GPU to GPU communication within a node. In V100-SXM2, each GPU has 6 NVLinks for bi-directional communication. The bandwidth of each NVLink is 25GB/s in uni-direction and all 4 GPUs within a node can communicate at the same time, therefore the theoretical peak bandwidth is 6*25*4=600GB/s in bi-direction. However, the theoretical peak bandwidth using PCIe is only 16*2=32GB/s as the GPUs cannot communicate in parallel. So in theory the communication with NVLink could be up to 600/32=18x faster than PCIe. To check how much improvement using NVLink over PCIe provides, we profiled the Peer-to-Peer (P2P) memory access time in three framework with Nvidia command line profiler nvprof [18]. The finding is that TensorFlow implemented P2P without explicit call to cudaMemcpyPeer() API. MXNet and Caffe2 achieved 3.7x and 3.2x speedup by using NVLink over PCIe among 4 GPUs in FP32 mode. For Caffe2, with 8 GPUs, its performance does not scale well compared to 4 GPUs and the reason has been explained in Section IV-A. Therefore, the improvement of SXM2 over PCIe is not as high as TensorFlow



(a) Horovod+TensorFlow

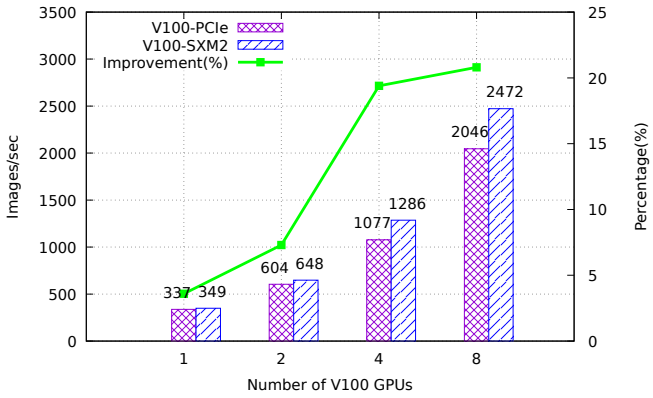


(b) MXNet

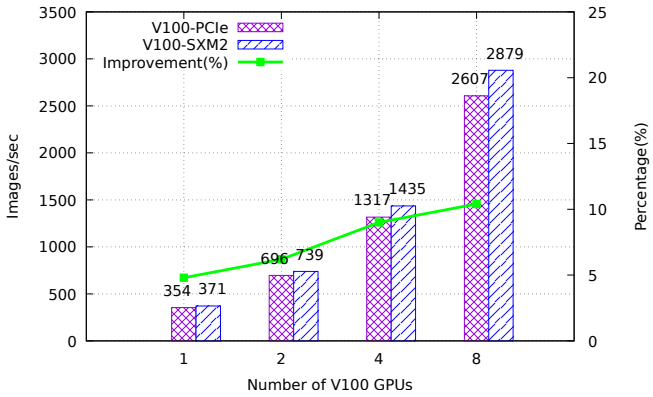


(c) Caffe2

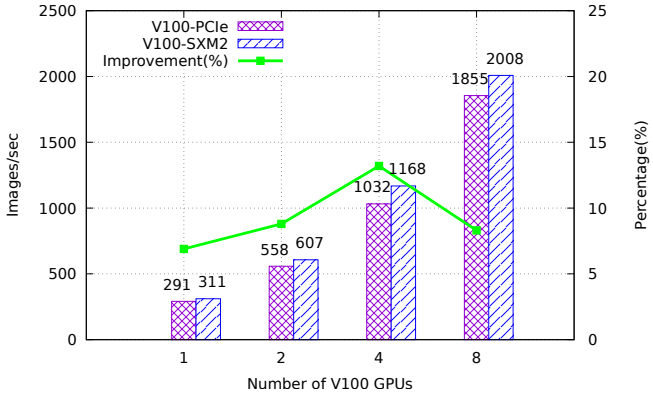
Fig. 2: The scaling performance for Resnet50 training using ILSVRC2012 dataset



(a) Horovod+TensorFlow



(b) MXNet



(c) Caffe2

Fig. 4: V100-SXM2 vs V100-PCIe for Resnet50 training with ILSVRC2012 dataset

and MXNet. Since the P2P memory accesses only take a small portion of the whole application time, as shown in Figure 4, the overall application execution does not improve significantly.

C. FP16 vs FP32

As mentioned in Section III, the V100 GPU includes Tensor Cores which support mixed precision training. Here we simply denote it as FP16 training. We doubled the batch size for FP16 tests since FP16 consumes only half the memory for floating points when compared to FP32. The performance comparison of FP16 versus FP32 is shown in Figure 5. The reason for the unstable result for Caffe2 has been explained in Section IV-A. Except for the distributed training in Caffe2, FP16 is around 60% to 107% faster than FP32.

D. V100 vs P100

To demonstrate the performance advantages of the Tesla V100 GPU over its previous generation P100 GPU, the per-

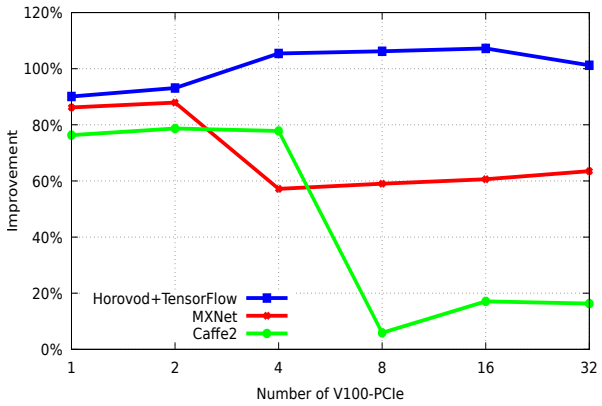


Fig. 5: The improvement of FP16 over FP32 for Resnet50 training with ILSVRC2012 dataset

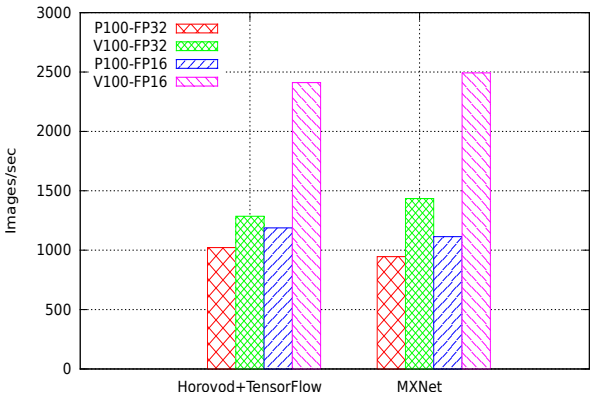


Fig. 6: V100 vs P100 for Resnet50 training with ILSVRC2012 dataset

formance of one node with four V100-SXM2 was compared to that of a node with four P100-SXM2. Figure 6 shows this performance comparison. The result shows that in FP32 mode, V100 is 26% faster than P100 when using TensorFlow, and 52% faster with MXNet. In FP16 mode, V100 is 103% faster than P100 with TensorFlow, and 123.8% faster with MXNet. The reason that V100 is much faster than P100 in FP16 mode is that P100 does not have Tensor Cores.

E. Storage Considerations

In our testing, a Dell PowerVault MD3220 [20] with 24 500GB NL-SAS 6Gb/s 2.5" hard drives were used. Two of the twenty four drives are hot spares and the rest are configured as a 9TB logical block device with RAID 6. This array was connected to the head node by a single 6Gbps SAS HBA and exported via NFS with IPoIB to the worker nodes. Since the ILSVRC 2012 dataset size is lower than the memory size in the compute nodes, to remove any caching effect in benchmarking, we clear the PageCache, dentries and inodes before each run. As a result, the performance data recorded in the first epoch does not have caching effects since this replicates the scenario where the data set is too large to fit into memory. This is a common occurrence in real world situations.

During the first epoch, the data is being pulled directly from disk as the cache is empty. This should also mimic scenarios in which the dataset is too large to fit into the cache of the storage server and therefore there will always be some subset of data reading from the disk in each epoch.

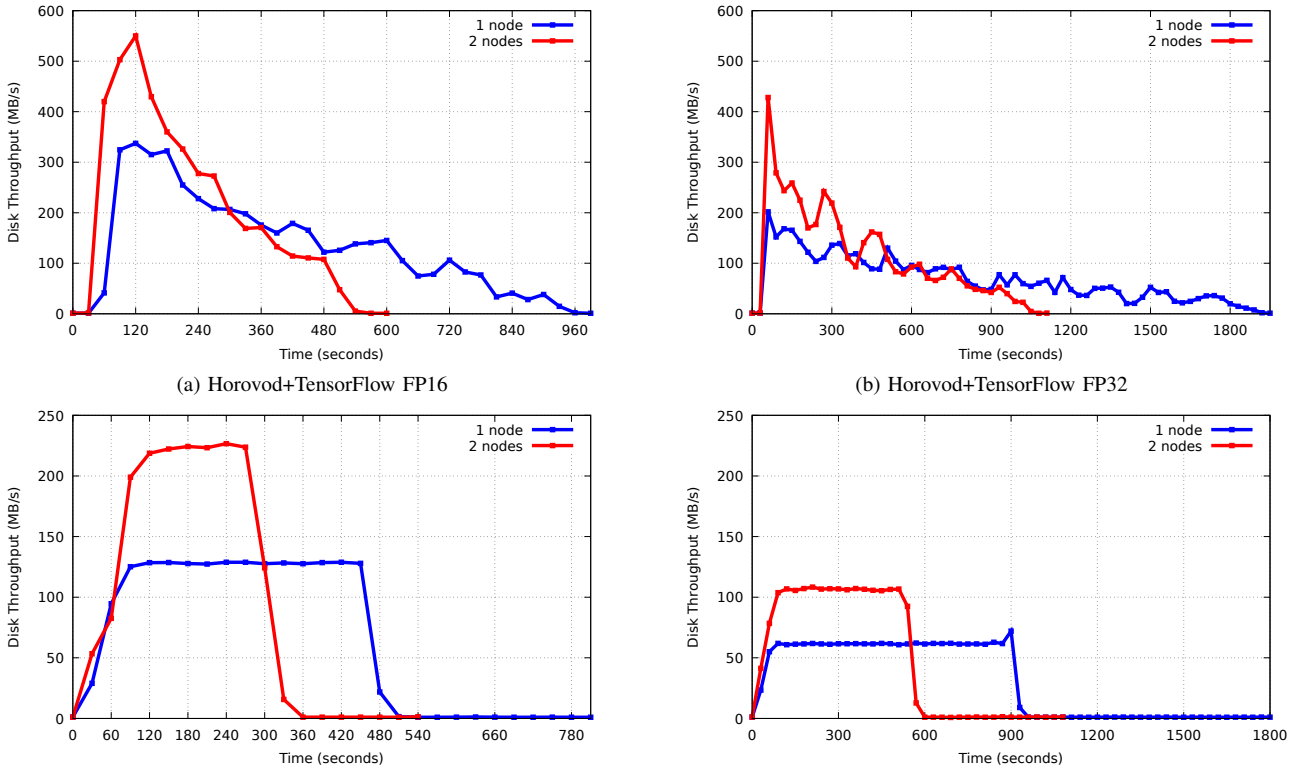
Storage Performance Snapshot 2018 [19] from Intel was chosen to be the storage profiling tool. It was run on head node to profile the aggregated performance of MD3220 storage as well as on all GPU compute nodes to provide individual insights on local disk and the network data movement.

Figure 7 shows the storage read performance with 1 and 2 compute nodes when running Resnet50 model with Horovod+TensorFlow and MXNet on up to two nodes with V100-SXM2 GPUs. There is almost no writing operations in deep learning training, so the throughput numbers only include disk reading throughput and they are depicted as the MB/s. Since running the neural network with two nodes is faster than using one node, the total time needed with two nodes is around half of using one node. The following conclusions can be obtained:

(1) There is positive correlation between storage read throughput and the number of compute nodes. The more nodes are used, the higher read throughput is observed. Consequently high end storage is needed when designing a large deployment. The peak throughput using two nodes is close to 2x compared to using one node.

(2) For MXNet, the read throughput is consistent through the first epoch, while for TensorFlow, the read throughput is decreasing through the first and second epoch. The reason is explained as follows. The ImageNet JPEG images are formated to 1024 TFRecords file database for TensorFlow and one RecordIO database file for MXNet. The database for both TensorFlow and MXNet is small enough to fit into main memory in each compute node. However, both frameworks behave differently in terms of storage throughput. In deep learning training, one epoch means traversing the whole dataset once. After each epoch, all files in the same dataset are shuffled. This technique is used for solving the data shortage issue. For MXNet, only one process is used for reading the only one database file, therefore the read throughput is consistent in the first epoch. In the second epoch, the whole dataset has been cached in the memory of each compute node, so the process does not need to fetch the same dataset from the disk anymore.

For Horovod+TensorFlow, although the whole dataset can fit into main memory, there is still read activity even after one epoch. This is because the Resnet50 training is implemented differently than MXNet. In Horovod+TensorFlow's implementation, four MPI processes are launched for four GPUs within a node. However, the four MPI processes do not read the distinct parts of the whole dataset. Instead they are given different random seeds for reading files, so there is high chance that they read some common files. As more and more common files are cached into compute nodes' main memory and storage system's cache, less files are needed to fetch from the disk.



(a) Horovod+TensorFlow FP16

(b) Horovod+TensorFlow FP32

(c) MXNet FP16

(d) MXNet FP32

Fig. 7: Storage read performance for Resnet50 training (2 epochs)

When the whole dataset is cached, there is no reading activities from the disks anymore. In the extream case when all four processes always read the same files, four epochs are needed to traverse the whole dataset. In practice, based on the results shown in Figure 7, two epochs are needed to traverse the whole dataset, as the read throughput become 0 at the end of the second epoch.

(3) Comparing the throughput number for FP32 and FP16, it is clearly shown that the same system running with lower precision mode requires more storage throughput. For instance, with two nodes, the peak throughput is ∼550 MB/s in FP16 mode, compared to only 430 MB/s in FP32 mode. The FP16 tests consistently read more data in TensorFlow and MXNet, because in all tests the batch size used in FP16 mode doubled compared to FP32 mode. This causes more image files to be prefetched in each iteration which made FP16 read operations 1.5 to 2 fold higher to FP32. These two different frameworks have their own database formats and their own way of prefetching data. But in general, the storage system needs to have fast read performance for applications optimized for tensor cores. Since FP16 reduced the memory footprint in GPU memory, this allowed more image files be loaded with larger batch size.

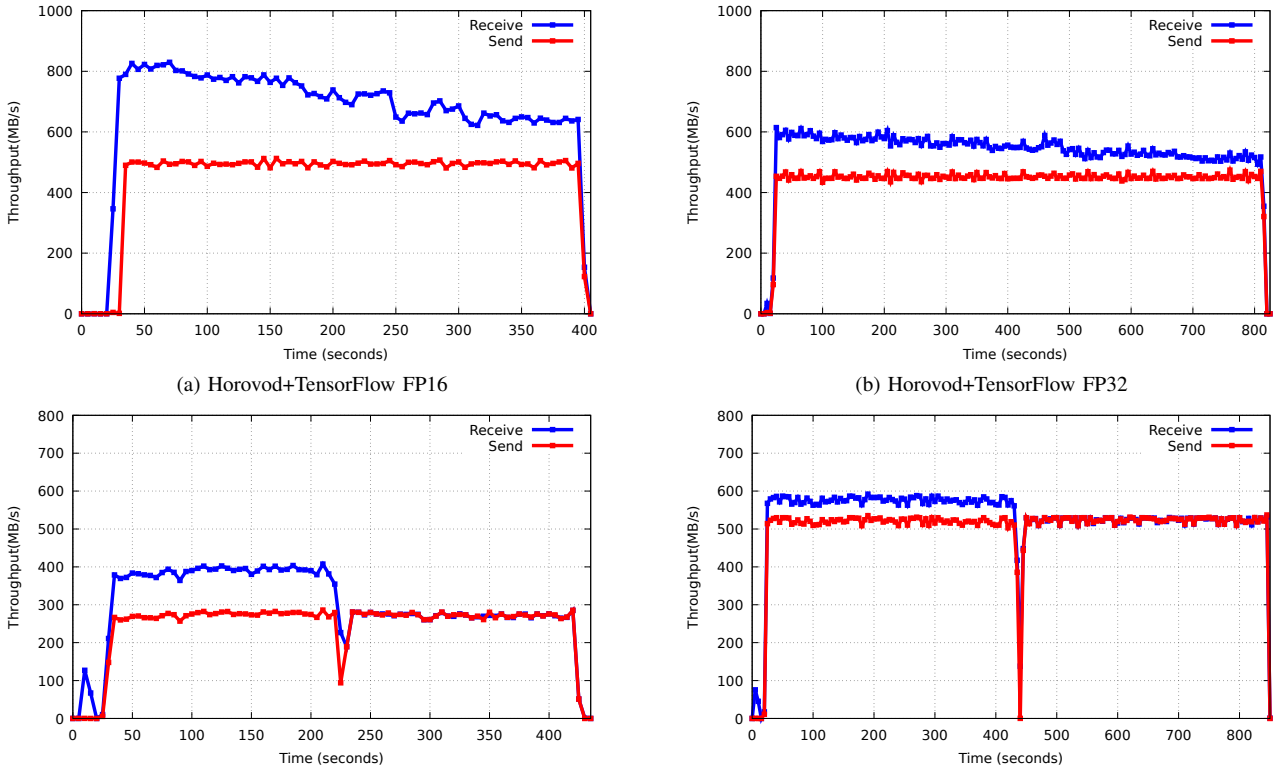
F. Node Interconnect Considerations

All multi-node tests were run on Mellanox 100Gbps EDR InfiniBand network. Each compute node has one InfiniBand EDR adapter, with Mellanox OFED 4.4 driver. As mentioned in the previous section, the MD storage array is directly connected to the head node and is shared as an NFS volume to all compute node via the 100Gbps EDR network.

As with the storage tests, caches have been cleared on head node and all compute node before testing began. The ImageNet dataset is not large enough while compared to real world production environment. If the caches didn't get cleared, the data will be buffered in head node or compute node's memory. Therefore clearing out all data in caches is meant to reflect the real world situation where data set is normally larger than what storage systems can cache.

The throughput (MB/s) over Infiniband EDR adapter was measured on all compute nodes with Mellanox Unified Fabric Manager [21]. Figure 8 shows the InfiniBand EDR network performance when running Resnet50 with both Horovod+TensorFlow and MXNet on two compute nodes. Both compute nodes have the same throughput numbers, so the performance in this figure is from only one compute node. The following conclusion can be reached:

(1) The receive throughput is higher than the send throughput. This is because two types of data are going through the InfiniBand network: the gradidents exchanging and the input dataset reading. In deep learning distributed training, different nodes work on their own dataset. After each iteration, the graidents calculated by all processes/threads in different nodes are reduced and then the reduced results are broadcasted



(a) Horovod+TensorFlow FP16

(b) Horovod+TensorFlow FP32

(c) MXNet FP16

(d) MXNet FP32

Fig. 8: InfiniBand network performance for Resnet50 training (2 epochs)

back to them. Therefore different nodes need to exchange gradients through InfiniBand network. Different nodes also read the input dataset through InfiniBand since the storage is mounted through InfiniBand. The receive throughput includes the throughput of gradient exchange between nodes and the input data, while the send throughput only includes the gradient exchange throughpupt.

(2) The gradients to be sent and received are the same for each compute node, and the difference between receive and send throughput is the disk read throughput. This can be verified together with the storage throughput in Figure 7. For instance, in TensorFlow FP16 mode, the receive and send throughput difference in the beginning of the training is ∼300 MB/s in one node. With two nodes reading data from the storage at the same time, the total disk throughput is ∼600 MB/s which matches the beginning peak number with 2 nodes in Figure 7. For MXNet, in the second epoch when no disk read operation happens, the receive througput line overlaps prefectly with the send throughput line.

(3) In FP16 training, the size of the gradients to be reduced is slightly higher than FP32 for Horovod+TensorFlow, but only half of FP32 for MXNet. This is because of their different implementations for FP16 training. The more detailed analysis and explanation from the algorithm level will be investigated in the future work. In MXNet, there is also a big drop after one epoch which is because of the shuffle operation for the whole dataset. The TensorFlow does not have this drop. This is because TensorFlow does not really shuffle the whole dataset. Instead it uses a buffer to sample from the whole dataset. The buffer size is 10,000 which is much smaller than the 1,281,167 images in the entire dataset.

V. Performance Tuning for DL Inference

Inference is the end goal of deep learning. The inference performance is also critical as it is either latency-focused or throughput-focused. This section quantifies the inference performance using NVIDIA's TensorRT library. TensorRT, previously called GIE (GPU Inference Engine), is a high performance deep learning inference engine for production deployments of deep learning models. It maximizes inference throughput and efficiency. TensorRT provides users the ability to take advantage of fast reduced precision instructions provided in the Pascal and Volta GPUs.

Although many HPC applications require high precision computation with FP32 (32-bit floating point) or FP64 (64-bit floating point), deep learning researchers have found they are able to achieve the same inference accuracy with FP16 (16-bit floating point) as can be had with FP32 [22]. Many applications only require INT8 (8-bit integer) or lower precision to keep an acceptable inference accuracy [23]. TensorRT began to support INT8 operations in version 2. All inference experiments were performed on a single V100-PCIe GPU.

Figure 9 shows the inference performance with TensorRT on Resnet50 model with different batch sizes. Note that a known

TABLE III: The accuracy between FP32 and INT8

Network	FP32		INT8		Difference	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
Resnet-50	72.90%	91.14%	72.84%	91.08%	0.07%	0.06%
ResNet-101	74.33%	91.95%	74.31%	91.88%	0.02%	0.07%
ResNet-152	74.90%	92.21%	74.84%	92.16%	0.06%	0.05%
VGG-16	68.35%	88.45%	68.30%	88.42%	0.05%	0.03%
VGG-19	68.47%	88.46%	68.38%	88.42%	0.09%	0.03%
GoogLeNet	68.95%	89.12%	68.77%	89.00%	0.18%	0.12%
AlexNet	56.82%	79.99%	56.79%	79.94%	0.03%	0.06%

issue is that on V100 GPU, running models with INT8 only works if the batch size is evenly divisible by 4 [24]. For this purpose we start with higher batch sizes. We can see that INT8 mode is ∼3.7x faster than FP32. This is expected since the theoretical speedup of INT8 is 4x compared to FP32 if only matrix multiplications are performed and no other overhead is incurred. However, there are kernel launches, occupancy limits, data movement and mathematical operations other than multiplications, so the speedup is reduced to about 3x faster.

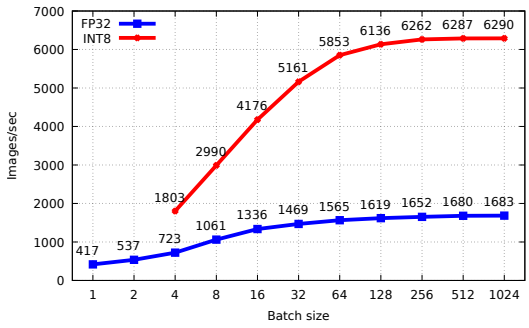


Fig. 9: TensorRT inference for Resnet50 with INT8 vs FP32

Deep learning inference can be applied in difference scenarios. In certain instances, delayed batch processing is acceptable in which case the focus is using large batch sizes to increase throughput. On the other hand, real-time scenarios are latency sensitive, with time to solution taking priority over efficient hardware utilization. In such cases, the batch size can be as small as 1. Therefore we also measured the performance difference when using different batch sizes as shown in Figure 9. It can be seen that without batch processing the inference performance is very low. This is because the GPU is not assigned enough workload to keep it busy. The larger the batch size was used, the higher the inference performance was produced. This begins to taper off as the batch size increases. The largest batch size is only limited by GPU memory.

We also compare the accuracy when using both INT8 and FP32 to verify that using INT8 can get comparable performance to FP32. To make INT8 data encode the same information as FP32 data, a calibration method is applied in TensorRT to convert FP32 to INT8 in a way that minimizes the loss of information. More details of this calibration method can be found in the presentation "8-bit Inference with TensorRT [25]. We used the ILSVRC2012 validation dataset for both calibration and benchmarking. The validation dataset has 50,000 images and was divided into batches where each batch has 25 images. The first 50 batches were used for calibration purpose and the rest of the images were used for accuracy measurement. Several pre-trained neural network models were used in our experiments, including ResNet-50, ResNet-101, ResNet-152 [26], VGG-16, VGG-19 [27], GoogLeNet [28] and AlexNet [29]. Both top-1 and top-5 accuracy were recorded using FP32 and INT8 and the accuracy difference between FP32 and INT8 was calculated. The result is shown in Table III. We can see the accuracy difference between FP32 and INT8 is between 0.02% - 0.18% for all test cases. This means very little accuracy is lost while achieving a 3x speed up.

VI. Conclusions and Future Work

In this paper, we performed a comprehensive benchmarking for Horovod+TensorFlow, MXNet and Caffe2 on a V100 GPU cluster. The benchmarking includes both the training and inference phases. For training, we compared the performance difference of single GPU vs multi-GPU, V100-SXM2 vs V100-PCIe, and FP16 vs FP32. We profiled those frameworks to understand the utilization of CPU, memory, network and storage and analyzed the reason for the unexpected performance for Caffe2. We found that for Caffe2, when the dataset is not fully cached into system's memory, the training performance is very unstable across iterations in multi-node training. For TensorFlow and MXNet, the differences for the database format and shuffle implementation lead to different storage and network behavior. The user is also able to infer the requirement of both storage and network based on our profiling results, for Resnet50 model and ILSVRC 2012 workload.

For inference, we benchmarked the throughput and accuracy with FP32 and INT8 using TensorRT library. As a result, INT8 is 3.7x faster than FP32 but can still achieve comparable accuracy.

In the future work, we will try larger data sets so that it cannot be fit into memory and then check the impact of different file systems. We will also try to experiment and analyze the model parallelism implementations for Recurrent Neural Networks. The Resnet50 model benchmarked in this paper is implemented with data parallelism in all frameworks. We would like to compare these two types of implementations. More storage systems and the latest 32GB V100 GPUs will also be evaluated in the future work.

REFERENCES

[1] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "TensorFlow: A System for Large-Scale Machine Learning," in *OSDI*, vol. 16, 2016, pp. 265–283.

[3] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *arXiv preprint arXiv:1512.01274*, 2015.

[4] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[5] S. Shi, Q. Wang, P. Xu, and X. Chu, "Benchmarking State-of-the-Art Deep Learning Software Tools," in *Cloud Computing and Big Data (CCBD), 2016 7th International Conference on*. IEEE, 2016, pp. 99–104.

[6] F. Seide and A. Agarwal, "CNTK: Microsoft's Open-source Deep-learning Toolkit," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 2135–2135.

[7] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "PyTorch: Tensors and Dynamic Neural Networks in Python with Strong GPU Acceleration," 2017.

[8] C. Coleman, D. Narayanan, D. Kang, T. Zhao, J. Zhang, L. Nardi, P. Bailis, K. Olukotun, C. Ré, and M. Zaharia, "DAWNBench: An End-to-End Deep Learning Benchmark and Competition," *Training*, vol. 100, no. 101, p. 102, 2017.

[9] A. Sergeev and M. Del Balso, "Horovod: Fast and Easy Distributed Deep Learning in TensorFlow," *arXiv preprint arXiv:1802.05799*, 2018.

[10] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour," *arXiv preprint arXiv:1706.02677*, 2017.

[11] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, "ImageNet Training in Minutes," in *Proceedings of the 47th International Conference on Parallel Processing*. ACM, 2018, p. 1.

[12] X. Wang, H. Zhao, and J. Zhu, "GRPC: A communication cooperation mechanism in distributed systems," *ACM SIGOPS Operating Systems Review*, vol. 27, no. 3, pp. 75–86, 1993.

[13] "Gloo Library," https://github.com/facebookincubator/gloo, 2018.

[14] J. L. Carlson, *Redis in Action*. Manning Publications Co., 2013.

[15] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic Differentiation in PyTorch," 2017.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A Large-scale Hierarchical Image Database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] "Nvidia Profiler User's Guide," http://docs.nvidia.com/cuda/profiler-users-guide/index.html, 2018.

[19] "Intel Storage Performance Snapshot 2018," https://software.intel.com/sites/products/snapshots/storage-snapshot, 2018.

[20] "DELL PowerVault MD3200/MD3220 Series of Storage Arrays," http://i.dell.com/sites/content/business/solutions/engineering-docs/en/Documents/powervault-md3200-md3220-technical-guidebook-en.pdf, 2018.

[21] "Unified Fabric Manager Software," http://www.mellanox.com/related-docs/prod_management_software/PB_UFM_Software.pdf, 2018.

[22] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep Learning with Limited Numerical Precision," in *International Conference on Machine Learning*, 2015, pp. 1737–1746.

[23] P. Gysel, M. Motamedi, and S. Ghiasi, "Hardware-oriented Approximation of Convolutional Neural Networks," *arXiv preprint arXiv:1604.03168*, 2016.

[24] "TensorRT Release Notes," http://docs.nvidia.com/deeplearning/sdk/tensorrt-release-notes/rel_3-0-1.html, 2018.

[25] "8-bit Inference with TensorRT," http://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf, 2017.

[26] "Deep Residual Networks," https://github.com/KaimingHe/deep-residual-networks, 2018.

[27] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-scale Image Recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[28] "BVLC GoogleNet Model," https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet, 2018.

[29] "BVLC AlexNet Model," https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet, 2018.