Evaluating On-Node GPU Interconnects for Deep Learning Workloads

Nathan R. Tallent[1], Nitin A. Gawande[1], Charles Siegel[1], Abhinav Vishnu[1], and Adolfy Hoisie[2]

[1] Pacific Northwest National Laboratory, Richland, WA, USA
`firstname.lastname@pnnl.gov`
[2] Brookhaven National Laboratory, Upton, NY, USA
`ahoisie@bnl.gov`

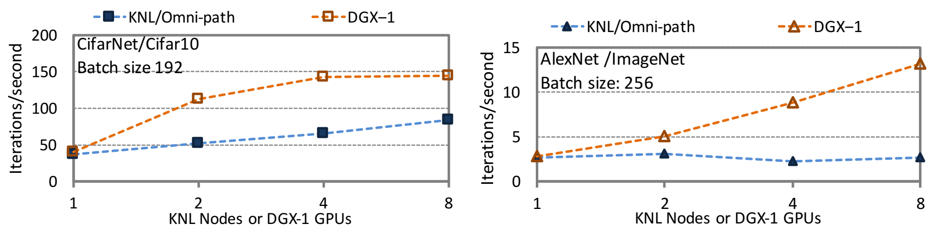
Abstract. Scaling deep learning workloads across multiple GPUs on a single node has become increasingly important in data analytics. A key question is how well a PCIe-based GPU interconnect can perform relative to a custom high-performance interconnect such as NVIDIA's NVLink. This paper evaluates two such on-node interconnects for eight NVIDIA Pascal P100 GPUs: (a) the NVIDIA DGX-1's NVLink 1.0 'hybrid cube mesh'; and (b) the Cirrascale GX8's two-level PCIe tree using dual SR3615 switch risers. To show the effects of a range of neural network workloads, we define a parameterized version of the popular ResNet architecture. We define a workload intensity metric that characterizes the expected computation/communication ratio; we also locate AlexNet and GoogLeNet within that space. As expected, the DGX-1 typically has superior performance. However, the GX8 is very competitive on all ResNet workloads. With 8 GPUs, the GX8 can outperform the DGX-1 on all-to-all reductions by 10% for medium-sized payloads; and in rare cases, the GX8 slightly outperforms on ResNet.

Keywords: GPU interconnects, NVIDIA DGX-1, NVIDIA NVLink, Cirrascale SR3615 switch riser, Convolutional Neural Networks

1 Introduction

Scaling deep learning workloads across multiple GPUs has become increasingly important in data analytics. For example, strong scaling can reduce the training time of neural networks. Moreover to train deep networks on large data sets, it may be necessary to harness multiple GPU memories.

The inter-GPU network can dictate performance when scaling deep learning workloads across multiple GPUs. Figure 1 shows that scaling some workloads is impossible without a high-performance interconnect [1]. The figure shows strong scaling behavior of two well known workloads — CifarNet/Cifar10 and AlexNet/ImageNet — on an NVIDIA DGX-1 [2] and an Intel Knights Landing [3] (KNL) cluster. The DGX-1 uses an NVLink-based GPU interconnect. The KNL cluster interconnects KNL processors (1 per node) using Intel's Omni-Path. For



(a) CifarNet/Cifar10 (b) AlexNet/ImageNet

Fig. 1: Performance scaling of (a) CifarNet/Cifar10 and (b) AlexNet/ImageNet on an NVIDIA DGX-1 and an Intel KNL/Omni-Path cluster.

each workload, the single-KNL/GPU performance is *very similar* — despite the GPU's higher peak floating point rate. However, scaling behavior is quite different. Although both workloads perform better over NVLink than Omni-Path, the qualitative scaling trends are different. With NVLink, the AlexNet workload (Fig. 1b) scales better than the CifarNet one (Fig. 1a). With Omni-Path, the qualitative scaling performance is *inverted*: scaling is better with CifarNet than AlexNet. The reason is that AlexNet's much larger all-to-all reduction operations (allreduce) place a much higher stress on interconnect bandwidth. Omni-Path, designed as a cluster interconnect, has a per-node (uni-directional) bandwidth of 12.5 GB/s whereas the DGX-1's NVLink supports up to 80 GB/s per GPU.

Because GPU interconnect performance can be a bottleneck when scaling deep learning workloads, some computing vendors are creating products to enable scalable GPU computing on a single densely populated node. A key question is how well a PCIe-based GPU interconnect can perform relative to a custom high-performance interconnect such as NVIDIA's NVLink. Unfortunately, it is difficult for data scientists to quantify the potential of these different products. In particular, Figure 1 shows that a high-performance interconnect *may not* be critical to scaling. The interconnect's importance depends significantly on a workload's characteristics, including total work and effective communication to computation ratio.

This paper evaluates two recent GPU interconnects (§2) for eight NVIDIA Pascal P100 GPUs on a single node: (a) the NVIDIA DGX-1's 'hybrid cube mesh' based on NVLink 1.0; and (b) the Cirrascale GX8's [4] two-level PCIe tree using two Cirrascale SR3615 switch risers.

We evaluate the two interconnects on a parameterized neural network workload (§3). The performance scaling of a parameterized neural network space has not been well studied. Other performance evaluations select specific networks — for example AlexNet [5] and GoogLeNet [6] — that have been designed for classifier performance, not workload evaluation. We define a parameterized variant of the popular ResNet [7] with controllable computational and communication intensities. With our parameterized ResNet, we show the effects of different neural network topologies and batch sizes on a workload's communication/computation ratio and scaling behavior. We define a workload intensity metric to characterize space of workload intensities and locate AlexNet and GoogLeNet within that space.

Our findings (§4) are as follows. The workload intensity metric in helpful in explaining scaling behavior. Given that the DGX-1's NVLink interconnect has more links and higher per-link bandwidth than the GX8's PCIe bus, it is not surprising that the DGX-1 typically has superior performance. However, we find that the GX8 is very competitive for all ResNet-style workloads; in rare cases, the GX8 slightly outperforms. Surprisingly, with 8 GPUs, the GX8 can outperform the DGX-1 on an allreduce benchmark by as much as 10% on payloads between 0.5 – 6 MB. In contrast, with 4 GPUs the DGX-1 allreduces outperform the GX8 by 40%. The reason is that with 8 GPUs, the PCIe network saturates more quickly with respect to payload size. The DGX-1 has a distinct scaling advantage for the communication-intensive AlexNet where we hypothesize that load imbalance enables its NVLink interconnect to perform closer to the 4 GPU bandwidths than 8, resulting in a 36% DGX-1 advantage.

2 Multi-GPU Computing Systems

This section describes the NVIDIA DGX-1 (Pascal) [2] and the Cirrascale GX8 (NVIDIA Pascal) [4] computing systems and then explains the test configuration. To isolate the interconnects, we configured the systems as closely as possible except for GPU interconnect.

Each system has a very similar host processor configuration. Both systems have a dual-processor host based on Intel Xeon processors. For the DGX-1, each processor is an Intel Xeon E5-2698v4; for the GX8, it is an E5-2697v4. The DGX-1's Xeon has 20 cores, two threads enabled per core, running at 2.2/3.6 GHz; and a 50 MB L3 cache, a 256 KB L2 cache shared between two cores, and 64 KB L1 cache per core. The GX8's Xeon has 18 cores with 2 thread/core, running at 2.3/3.6 GHz; L3 45 MB. In both cases, host memory is 512 GB DDR4-2133. Both systems use PCIe 3.0.

All important workload activities (e.g., neural network training) occurs on the GPUs. The primary work the host CPU performs is reading the initial training data set into memory and transferring it to the GPUs. Both systems read the large training inputs files from a local SSD whose throughput is sufficient to overlap training and reading.

2.1 NVIDIA P100 Pascal

Both systems have eight NVIDIA Tesla P100 (Pascal) GPUs. To isolate the interconnects, we configured the systems with the closest possible GPUs: Tesla P100-SXM2 and P100-PCIE-16GB. The DGX-1 has the former and the Cirrascale the latter. The only P100 available with NVLink support is the P100-SXM2; and because of NVLink support it uses a different form factor (SXM2). The P100-PCIE-16GB is the 'highest bin' P100 available with the PCIe 3.0 $\times 16$ interface. The only differences between the two P100s — besides NVLink and form factor — are SM clock speed (1328 vs. 1189 MHz) and TDP (300 vs. 250 Watts).

Pascal GPUs are fabricated with a 16 nm process. Each GPU has 3584 CUDA cores divided into 56 streaming multiprocessors (SM), where each SM has 64 CUDA cores. The P100-PCIE-16GB has a peak FP performance of 9.3 Teraflops single precision (4.67 Teraflops double). Due to the higher clock rate, the P100-SXM2 has a peak FP performance of 10.6 Teraflops single precision (5.3 Teraflops double). Each GPU has 16 GB high-bandwidth global memory (HBM2), a 4096-bit memory bus operating at 715 MHz (split into 8 memory controllers), and 4 MB L2 cache.

Normalizing GPU Performance Given the different GPUs, it is necessary to distinguish the performance effects of the varying GPU clocks from the different interconnects. One possibility is normalizing or scaling GPU performance post facto. This approach is difficult with fixed clocks; and more difficult with dynamically boosted clocks. Rather than attempting this approach, we power-capped both GPUs. The obvious approach is to cap both GPU variants at the nominal frequency of the P100-PCIE-16GB, 1189 MHz. To present results as close to the P100-SXM2 as possible, we found the maximum sustained frequency of the P100-PCIE-16GB for a representative workload. That is, we empirically identified the maximum frequency for the P100-PCIE-16GB to execute without throttling. Based on this study, we capped both GPUs at 1227 MHz, which closes the gap by 27%. With this experimental setup, we expect the performance of each GPU to be identical. The GPU performance is still sufficiently high to highlight the scaling effects of each interconnect.

2.2 NVIDIA DGX-1 and NVLink 1.0

Figure 2 shows the DGX-1's intra-node interconnect topology [2]. Each GPU's SXM2 interface, in contrast to the more conventional PCIe interface, connects directly to the NVLink interconnect. The NVLink interconnect enables intra-node GPU communication. Each GPU has 4 NVLink lanes arranged in a 'hybrid cube mesh' topology. The hybrid cube mesh has two directly connected groups of 4 along with 3D hypercube links between the groups. The topology ensures that a GPU is no more than two hops away from another GPU.

Each of the 4 NVLink lanes supports 20 GB/s in both directions. Thus, the total NVLink uni-directional bandwidth of a GPU is 80 GB/s. Each GPU also connects via a PLX-switch to a PCIe 3.0 $\times 16$ bus with maximum bandwidth of 16 GB/s (uni-directional). This PLX switch serves as a connecting point between GPUs and CPUs, and a potential InfiniBand network.

2.3 Cirrascale GX8 and SR3615 Switch

The Cirrascale GX8 [4] system supports direct communication between 8 GPUs using two Cirrascale SR3615 switch risers [8]. Communication occurs over the PCIe bus, enabling a single memory address space.

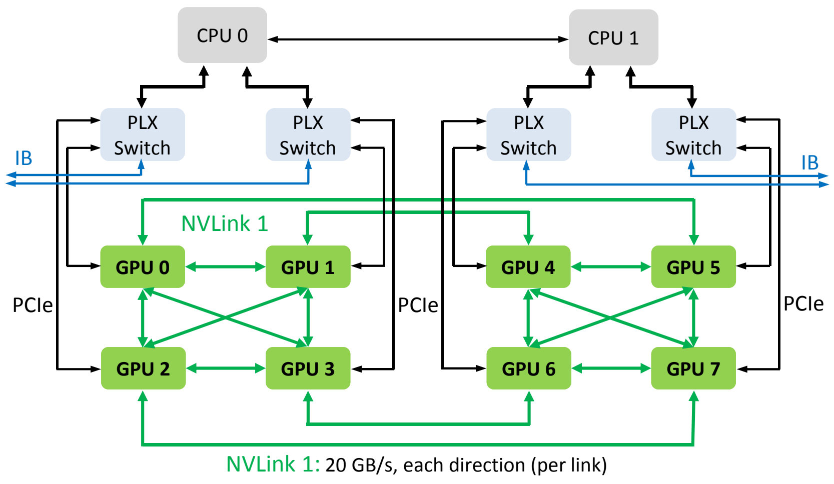


Fig. 2: Inter-GPU network on NVIDIA DGX-1.

Figure 3 shows the GX8's inter-GPU network. To enable communication over a single PCIe bus (and hence single memory address space), the GX8 uses a tree topology rooted at only *one* of the host CPUs [9]. The two-level tree is rooted at one host's on-die PCIe controller, a.k.a. the root complex, supporting PCIe 3.0 $\times 40$. Attached to that host CPU are two SR3615 switch risers. Each SR3615's upstream is PCIe 3.0 $\times 16$ (16 GB/s uni-directional). Two risers consume 32/40 lanes of the root complex. Communication between the SR3615s occurs via the root complex using the standard PCIe bus.

Four P100s are attached to each SR3615 switch riser. Each GPU (P100-PCIE-16GB) has a PCIe 3.0 $\times 16$ interface. Thus, each switch riser's input is 64 PCIe lanes of GPU; and 16 out. As a result there is a peak uni-directional 16 GB/s (PCIe 3.0 $\times 16$) between any two GPUs.

Because of the SR3615 switch, communication paths do not all need to traverse the root complex. A pair of GPUs attached to different risers traverse two switches and the PCIe root complex. However, a pair of GPUs attached to the same switch require no intermediate paths.

2.4 Inter-GPU Communication

For inter-GPU (peer-to-peer) communication, we use a combination of CUDA 8.0 and the NVIDIA Collective Communications Library (NCCL). CUDA 8.0 includes support for GPUDirect, or GPU-to-GPU direct memory access (DMA). NCCL [10, 11] is a library for inter-GPU collective communication and synchronization. NCCL's collective algorithms are based on topology-aware rings and optimized for throughput [12, 13]. NCCL is interconnect-aware and thus the same collective call uses, as appropriate, the NVLink or PCIe interconnect. Available collectives include allgather, allreduce, and broadcast.

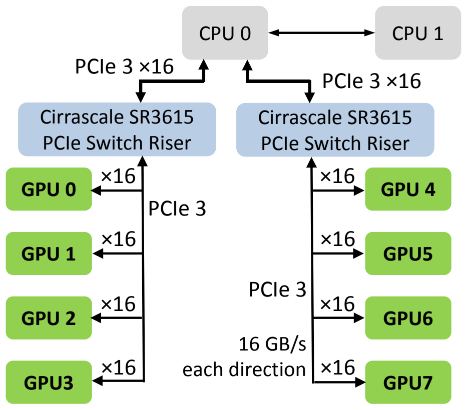


Fig. 3: Inter-GPU network on Cirrascale GX8.

To achieve high throughput on large payloads, NCCL's algorithms are pipelined based on small 4–16 KB chunks and GPUDirect peer-to-peer direct access. With large payloads, pipelining hides the linear latency term of the ring resulting in transfer bandwidths approaching link bandwidth [14]. However, for small messages, the ring latency is exposed.

3 Workloads

In this paper, we develop a systematic approach for characterizing and specifying neural network workloads. To explore the effects of different neural network topologies and batch sizes on scaling behavior, we define a parameterized variant of the popular ResNet [7] with controllable computational and communication intensities. We complement our study with results from the well known AlexNet [5] and GoogLeNet [6]. The subsections below describe each CNN architecture. After each network is described, we characterize the space of workload intensities and locate AlexNet and GoogLeNet within that space.

Each distinct neural-network training workload executes in the following manner. First, a given neural network architecture is replicated on each GPU. Then, the neural network is trained, processing an image dataset sequentially in batches or *iterations*. For each batch, images are divided among available GPUs for data parallelism. To train, each GPU processes its images resulting in a series of model activations — floating point operations — resulting in distinct values for each GPU's copy of model parameters. At the end of each iteration, allreduce operations ensure each GPU's model has an identical copy of model parameters.

For all workloads, we use the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [15], a well known benchmark for object classification and detection. Specifically, we use ILSVRC2012 which has 1000 object classes and 1.43 M images annotated images, each of size 256×256.

3.1 AlexNet

AlexNet [5] uses the ImageNet (ILSVRC2012) [15] dataset. Compared to non-deep learning methods, AlexNet has performed well on ILSVRC2012. AlexNet has five convolution layers, three pooling layers, and two fully-connected layers. This CNN architecture requires about 1.4 M activations/image and has 60 M parameters.

3.2 GoogLeNet

GoogLeNet [6] is more complex model than AlexNet. GoogLeNet has two convolution layers, two pooling layers, and nine inception layers. Each inception layer consists of six convolution layers and one pooling layer. The concept of inception layer is to cover bigger area of images while maintaining fine resolution for small information on these images. The inception module of GoogLeNet concatenates filters of different sizes into a single new filter. This avoids parameter explosion with the use of inception layers. GoogLeNet performs significantly better than AlexNet for the ImageNet and the recent ILSVRC [15] challenge datasets. This CNN architecture has about 5.5 M parameters. GoogLeNet in relation to AlexNet has (i) more layers; (ii) fewer features per layer, and; (iii) more activations. GoogLeNet has 10.8 M activations per image.

3.3 ResNet/x

Deep Residual Learning Network (ResNet) [7] introduced the concept of a residual block. Each block consists of two convolution layers along with a connection adding the output of the second block to the input of the first. Residual blocks are designed to allow the training of substantially deeper models than had been trained previously. By adding the input of the block to its output, the residual block learns the *residual* function, and forwards the activations to deeper layers than earlier. One advantage of ResNet is that it can improve accuracy of the model while avoiding parameter explosion. That is, the ResNet blocks increase the depth (and inner layers) of the network instead of its width.

Using residual blocks as a fundamental building block, several ResNet incarnations have been designed by researchers, including ResNet50 and ResNet1000. ResNets of various depths outperform GoogLeNet on the ILSVRC challenge, with a 50 layer network — consisting of a convolutional layer, 48 residual blocks, and a classifier layer — winning in 2015.

To explore the effects of different ResNet networks, we generate several ResNet variants by defining each network's inner layers to be a multiple of a 'ResNet block'. This enables us to explore how neural network topology and training batch size affects its communication/computation ratio and scaling. We define ResNet/x to be a standard ResNet input and output layer but where the inner layers are defined by x replications of the 'ResNet block'. Thus, ResNet/1 is a single convolution layer followed by a residual block and finally a classifier layer. Similarly, ResNet/16 has the same convolution and classifier layers as ResNet/1 but 16 residual blocks. Using this parameterized definition, we can explore the different computation and communication ratios by simply increasing the depth of residual blocks.

Each ResNet block has a certain number of features. As a result, increasing ResNet blocks proportionally increases activations/image and model parameters. More precisely, activations/image as a function of the block replications x is given with the following expression: $1,204,224x + 11,55,113$. Similarly model parameters as a function of replications is given by $46,211x + 74,857$. Thus, our ResNet/x models have the activations/image and parameters shown in Figure 4.

x	**Activations**	**Parameters**
1	2.4M	121K
2	3.6M	167K
4	6.0M	260K
8	10.8M	445K
16	20.4M	814K
32	39.7M	1,554K

Fig. 4: Activations and parameters for ResNet/x.

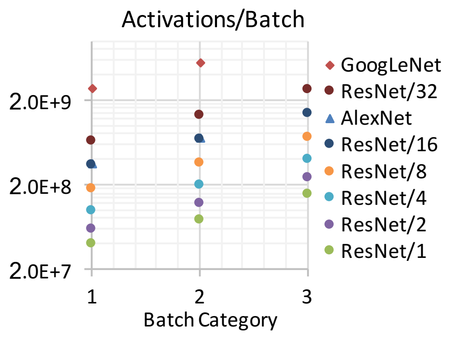
3.4 Workload Characterization

Figure 5 overviews the workloads we used in our study. To leverage well-known, verified, and optimized implementations of convolutional neural networks (CNN), we based our experiments on Convolutional Architecture for Fast Feature Embedding (Caffe) framework [16, 17], a widely used framework for CNN models. Caffe is a collection of state-of-the-art deep learning algorithms and reference models in a clean and modifiable framework accessible through a open source repository [18].

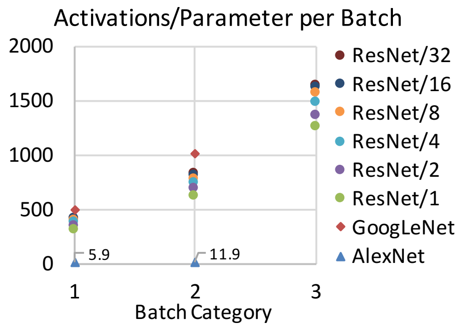
Figure 6 characterizes each workload's batch properties using metrics representing work and work intensity. Fig. 6a shows activations per batch, a measure of total GPU work. The horizontal axis refers to the batch categories in Fig. 5. (AlexNet and GoogLeNet each have two categories while ResNet/x has three.) Observe the large spread of work shown along the vertical axis (independent

CNN	**Dataset**	**Caffe Model**	**Batch Sizes**
AlexNet	ImageNet	bvlc_AlexNet	256, 512
GoogLeNet	ImageNet	bvlc_GoogLeNet	256, 512
ResNet/x	ImageNet	custom ResNet, $x \in \{1, 2, 4, 8, 16, 32\}$	16, 32, 64

Fig. 5: CNN architecture models and input datasets.



(a) Work per Batch



(b) Work Intensity per Batch

Fig. 6: Each workload's (a) work and (b) work intensity (work/communication).

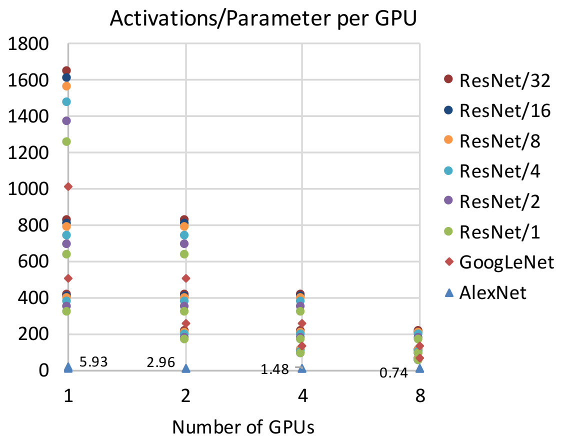
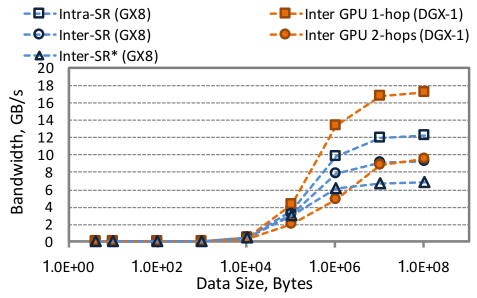


Fig. 7: Each workload's intensity (work/communication) during strong scaling.

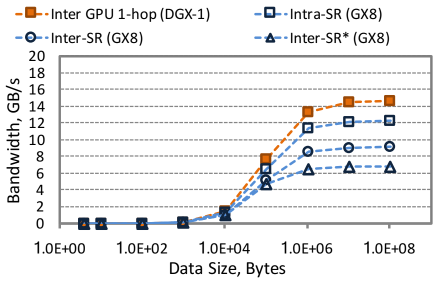
of the horizontal axis). The points densely cover over two orders of magnitude, specifically between 38M and 5,500M activations/batch.

Next we characterize *work intensity*, a measure of the ratio of communication to computation. Fig. 6b shows activations per parameter for each batch, a measure of the batch's work intensity. We capture well over two orders of magnitude of intensities, between 6–1650 activations/parameter. Our ResNet/x parameter sweep densely covers the space between 300–1650; and it sandwiches GoogLeNet.

Finally, we characterize each execution's *work intensity*. For each performance experiment, the batch's work is strong-scaled across 1, 2, 4 or 8 GPUs. Figure 7 shows activations per parameter for each GPU, a measure of the communication/computation ratio during execution. We capture well over three orders of magnitude of intensities, between 1–1650 activations per parameter per GPU. Our ResNet/x parameter sweep densely covers most of the space (between 40–1650); again, it sandwiches GoogLeNet.



(a) GPU-to-GPU memory copy



(b) GPU-to-GPU DMA

Fig. 8: Bandwidth of GPU-to-GPU memory copy for DGX-1 and GX8.

4 Evaluation

We conduct a performance evaluation using strong scaling to highlight effects of interconnect performance. Strong scaling is often desirable to reduce response time. With strong scaling, the amount of available per-GPU work systematically decreases, increasing the communication to computation ratio. In contrast to strong scaling, weak scaling tends to mask performance effects of weaker interconnects.

We used NVIDIA's optimized Caffe, a fork from BVLC-Caffe [18] optimized for the DGX-1 architecture [19]. For AlexNet and GoogLeNet, we used NVIDIA's provided models. For ResNet/x, we defined custom versions. We confirmed that all executions produced semantically meaningful results in that the models were equivalent to a sequentially equivalent execution.

We present our results in four subsections. The first two subsections discuss microbenchmarks for inter-GPU copies and NCCL collectives. We then show scaling results for AlexNet and GoogLeNet. Finally we discuss ResNet/x.

4.1 Inter-GPU Data Transfer

We used MGBench [20] to collect bandwidths and latencies between pairs of GPUs for GPU-to-GPU memory copy and GPU-to-GPU DMA (direct memory access).

Figure 8a shows bandwidths between pairs of GPUs for GPU-to-GPU memory copy. (Units are in power of 2, or GiB.) This unidirectional GPU-to-GPU memory copy is pipelined using CUDA's asynchronous memory-copy primitive. Rather than showing the full matrix for all pairs, we group the results by value clusters, where each group has an insignificant spread.

Figure 9 shows latencies of GPU-to-GPU memory copy highlighted at four different data sizes. A horizontal axis label of x-y means GPU x sent data to GPU y. Although the figure shows data with GPU 0 as source, we validated that using other GPUs as source produced qualitatively similar results.

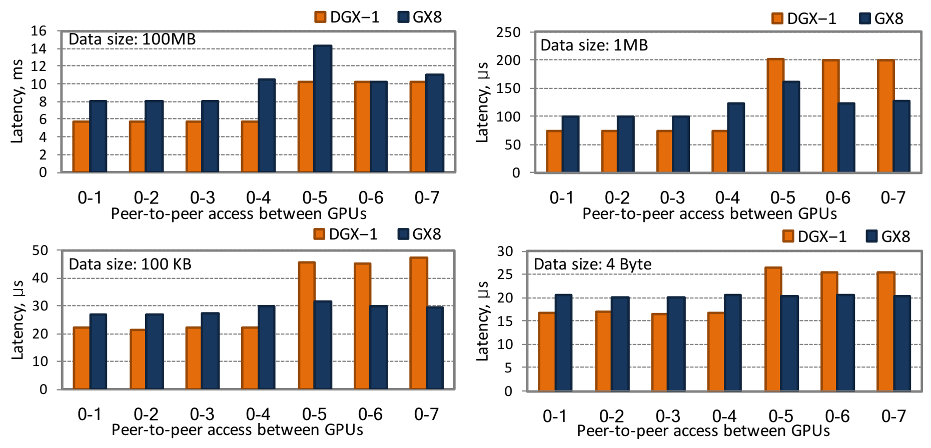
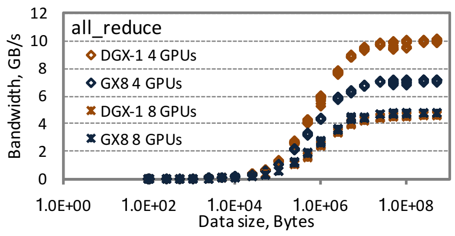


Fig. 9: Latency of GPU-to-GPU memory copy for DGX-1 and GX8.

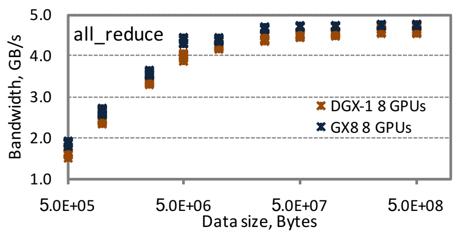
For both figures, the DGX-1 results are typically clustered in two groups, one representing a single NVLink hop and the other representing two NVLink hops. The *one-hop* data corresponds to communication within a fully-connected 4-GPU cluster; achieved bandwidth is about 85% (17.2 GB/s) of the 20 GB/s per-link peak. The *two-hop* data corresponds to communication between 4-GPU clusters; achieved bandwidth is about 50% (9.6 GB/s) of the peak.

The GX8 results are clustered in three groups. The groups are clearly seen in the latency plots (Fig. 9) for payload sizes 1 MB and above. The first two groups, *Intra-SR* and *Inter-SR*, correspond to communication within and between an SR3615 switch riser (SR), respectively. These groups are analogous to DGX-1 groups in that each SR forms a fully connected 4-GPU cluster. The Intra-SR achieved bandwidth is about 75% (12.2 GB/s) of peak (16 GB/s). The Inter-SR group includes GPUs 4, 6 and 7; achieved bandwidth is about 60% (9.6 GB/s) of peak. The third *Inter-SR** group captures the anomaly of sending data from GPU 0 to 5. It turns out that the second logical PCIe slot (GPU5) has longer physical signal paths between some elements on the ASIC which can lead to delays in dequeuing PCIe packets [21]. The circumstances in which these delays occur are narrow and more likely to originate within a microbenchmark than a real world application. For example, we do not observe the behavior in collective benchmarks.

Interestingly, the GX8 can have better bandwidth and latencies between GPUs that are in different 4-GPU-clusters. Compare Fig. 8a's Inter-SR and Iter-SR* GX8 curves with the 2-hop DGX-1 curve. The GX8's PCIe bandwidth saturates more quickly with respect to message size, resulting in higher GX8 bandwidth at medium-sized messages. Fig. 9 shows the same effect in terms of latency for message sizes 1 MB and below. In contrast, but as is expected, within a fully connected 4-GPU-cluster, NVLink's bandwidth and latency are better than PCIe.



(a) Effective bandwidth of allreduce



(b) Zoomed for typical payloads

Fig. 10: Effective bandwidth for NCCL allreduce on DGX-1 and GX8.

Another interesting comparison is that although NVLink latencies are very predictable, GX8 latencies are dependent on message size. The NVLink latencies fall into two clusters — one-hop and two-hops — independent of data size. In contrast, the GX8 latencies fall into 1 or 3 clusters depending on data size. For very small payloads (4 bytes), the GX8 latencies are flat and independent of hops. This shows the PCIe switching latency — even across multiple hops — is very low. The 1-cluster phenomenon largely holds true at 100 KB. By 1 MB, the GX8 results are clustered into the three groups described above. We hypothesize that the reason that small messages appear to be independent of topology — in contrast to NVLink — is related to PCIe switch buffering that effectively enables pipelining of smaller messages across multiple PCIe switches.

Figure 8b shows GPU-to-GPU bandwidths for DMA (direct memory access). The DMA data closely corresponds to the memory copy data. For DGX-1, we only shows single-hop DMA bandwidth because CUDA 8.0 does not support GPUDirect across 2 NVLink hops. In contrast, DMA is supported over a single PCIe bus.

4.2 Inter-GPU Collectives

Figure 10 shows *effective bandwidth* of NCCL allreduce, the key collective used in training. (Bandwidths are in power of 2, or GiB.) These results characterize allreduce performance given perfect GPU load balance. The size of allreduce payloads is the neural network's parameters represented as single precision floating points, or $4 \times$ *parameters* bytes. Thus for ResNet/x, payloads range from 0.5 – 6 MB and for GoogLeNet and AlexNet they are 22 MB and 240 MB, respectively.

We define effective bandwidth to be relative to a *single* GPU's payload, i.e., 1-GPU-payload/runtime. With this metric, the ideal value is relative to the bandwidth of one link. For example, in a fully connected 4-GPU NVLink cluster, the ideal allreduce is performed in 1 step where each GPU concurrently utilizes 3 links, yielding an effective bandwidth of 1 link, or 20 GB/s. Similarly, 8-GPU allreduce requires in the best case one more step, meaning the maximum effective bandwidth is halved to 10 GB/s. Three more considerations imply that

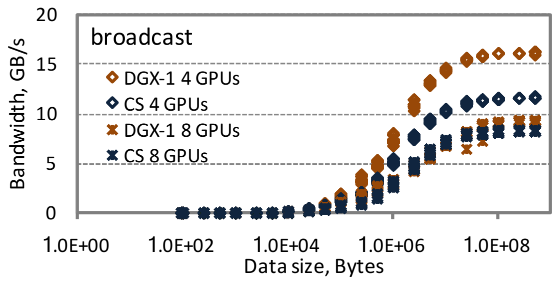


Fig. 11: Bandwidth for NCCL broadcast collective on DGX-1 and GX8.

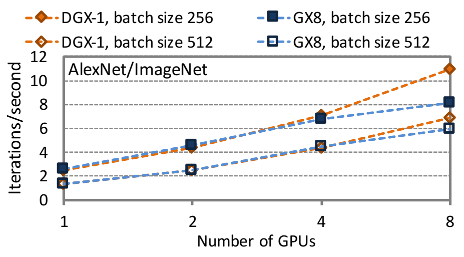
an allreduce's effective bandwidth should be less than ideal. As already observed, achievable link bandwidth for a direct copy is about 85% of ideal. Further, each GPU must execute the reduction operator. Finally, there is synchronization overhead.

Figure 10a shows that the NCCL allreduce is implemented well; there does not appear to be appreciable optimization headroom. The following observations explain. Within a fully connected 4-GPU-cluster, an allreduce's maximum effective bandwidth on DGX-1 and GX8 is 10 and 7 GB/s, respectively. For both systems, these values are about 60% of achievable bandwidth; they are also higher than *copying* data between inter GPU-clusters. This implies that NCCL's allreduces are effectively using a one-step algorithm. Between fully connected GPU-clusters, an allreduce's maximum effective bandwidth is 4.6 and 4.7 vs GB/s for DGX-1 and GX8, respectively. Given the extra hop, we assume that the maximum achievable bandwidth is half of the single link transfer bandwidth, or 8.6 and 6.1 GB/s. The above effective bandwidths are about 50% and 75% of maximum.

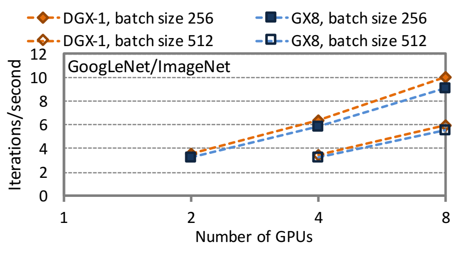
Interestingly, Figure 10a shows that the relative performance of allreduce depends on number of GPUs. For 4 GPUs, the DGX-1 has a 40% performance advantage for large messages (such as those used for AlexNet). For 8 GPUs, between fully connected GPU clusters, the GX8 has 3% better performance for large messages.

Figure 10b highlights allreduce's effective bandwidth on 8 GPUs for the payloads encountered in our ResNet/x workloads. The figure shows that the performance divergence between 0.5 and 5 MB payloads averages 10% in favor of the GX8. Clearly, as observed in GPU-to-GPU copies (§4.1), PCIe bandwidth saturates more quickly with respect to payload size. We expect that PCIe switching hardware is part of the explanation.

Finally, we observe that performance varies depending on collective. Collectives have two costs: data transmission and synchronization. An allreduce must synchronize with all GPUs using *all-to-all* synchronization. This synchronization attenuates the NVLink's potential advantages. We would therefore expect a single-root collective such as broadcast, where GPUs synchronize only with the root GPU, to have a difference performance profile. Figure 11 shows that on 8 GPUs, the DGX-1 does have slightly higher effective bandwidth.



(a) AlexNet/ImageNet



(b) GoogLeNet/ImageNet

Fig. 12: Strong-scaling (ImageNet): AlexNet and GoogLeNet on DGX-1 and GX8.

4.3 Strong Scaling of AlexNet and GoogLeNet

Figure 12 shows strong-scaling performance for AlexNet and GoogLeNet training on the DGX-1 and GX8. We collected results using two different batch sizes (256 and 512 images) on 1, 2, 4, and 8 GPUs. Although the batch sizes would be considered large for a single GPU, they are not large when scaling to 8 GPUs. Caffe data-parallelism distributes the images in each batch. Thus, with 256 batch size and 8 GPUs, there are 32 images per GPU.

Recall that we power cap GPUs to equalize the slightly different SM frequencies between the P100 SXM2 and PCIe variants. Therefore we expect both systems to have same single-GPU performance. As shown in Fig. 12a, this expectation is true for AlexNet. GoogLeNet's results (Fig. 12b) have data points missing for 1 and 2 GPUs. The reason is limited GPU memory capacity. With 1 and 2 GPUs, there was not enough memory to store both the GoogLeNet activations and the training data.

The most interesting results is that NVLink is far more important for AlexNet scaling than for GoogLeNet: for AlexNet the DGX-1 has a 36% advantage (11.0 vs. 8.1 iterations/s). It is more difficult than it might seem to explain the much higher DGX-1 performance on 8 GPUs. On one hand, as shown in Figure 7, AlexNet's activations/parameter per GPU is very small: past 4 GPUs, the metric is less than 1, meaning the workload is communication intensive. However, as noted in §4.2, the GX8 has slightly *higher* performance for 8 GPUs on an allreduce benchmark. Validating the root cause is difficult because of the limited value of GPU performance tools. Further more, to show the best performance results, we use NVIDIA's (read-only) Docker version of Caffe, which cannot be instrumented.

On GoogLeNet, the benefit of NVLink is comparatively small. As shown by Figure 6b, GoogLeNet is more compute intensive (in activations/parameter) by almost a factor of 100. Whereas AlexNet's intensities are 5.9 and 11.9 per batch category, GoogLeNet's are 500 and 1004, respectively.

Finally, NVLink becomes less important as batch size increases. This is not surprising as a larger batch size increases the per-GPU computation without changing communication, therefore reducing the importance of the interconnect.

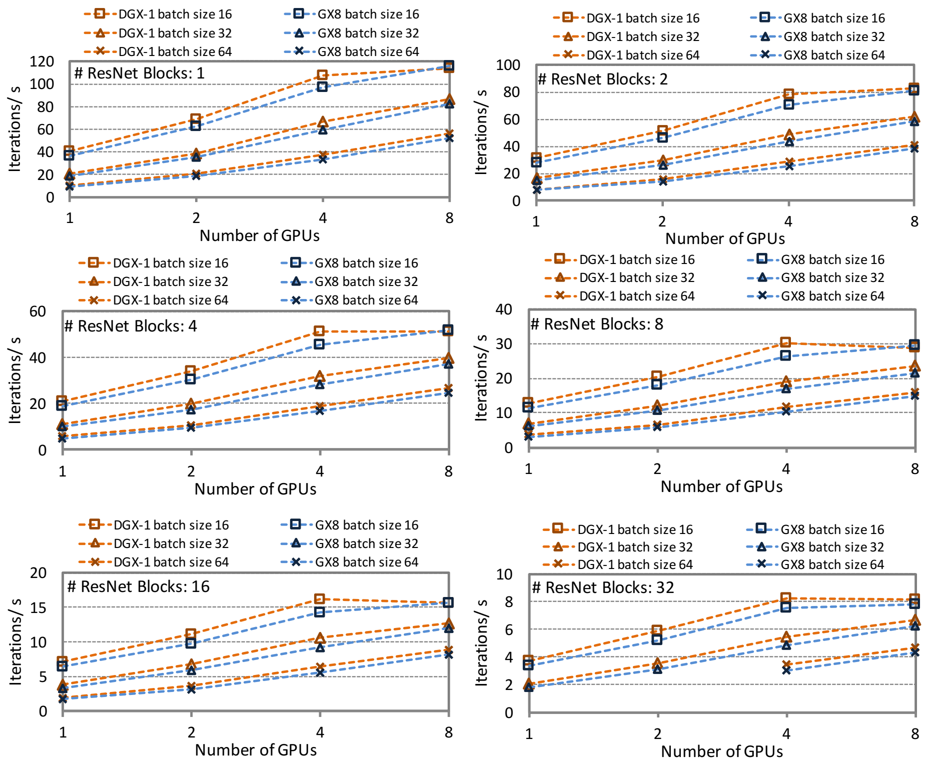


Fig. 13: Strong-scaling (ImageNet): ResNet/x on DGX-1 and GX8.

4.4 Strong Scaling of ResNet/x

Figure 13 shows strong-scaling performance for ResNet/x on the DGX-1 and GX8. We use smaller batch sizes for ResNet than with AlexNet or GoogLeNet. From a learning perspective, ResNet tends to use smaller batch sizes. Because of the deep network, a smaller batch size yields more updates per training epoch, which affects convergence. Also, the activations' memory consumption means larger network sizes will not fit in GPU memory. Observe that with ResNet/32, batch size 64 would not fit in the memory of either 1 or 2 GPUs.

Furthermore, the smaller batch sizes highlight GPU interconnect effects. Our custom ResNet/x has many fewer model parameters than either AlexNet or GoogLeNet, yielding smaller allreduce payloads and reducing the effects of inter-GPU communication and synchronization. We therefore compensate by using smaller batch sizes. The smaller batch sizes results in less per-GPU work, maintaining pressure on interconnect.

First, we discuss single-GPU performance. As before, we expect both systems to have same single-GPU performance given the power capping to equalize SM frequencies. Curiously, we see single-GPU performance converging as batch size increases. Thus, although the expectation holds true for batch size 64, there is a small divergence for batch sizes 16 and 32. We are not sure how to explain the divergence but have identified two possible factors.

One factor could be that for each batch of images, there is some CPU-based images processing overhead. For ResNet/x, this processing includes random horizontal flips, random crops, and subtraction of mean values to center distributions at 0. For smaller batch sizes, there is a potential this overhead can be exposed.

A second factor is host-CPU-based operations such as memory operations (e.g., `cudaMemset` and `cudaFree`) and scattering batch images to GPUs. This operations would occur over each system's PCIe bus. Although both host-to-GPU PCIe busses are PCIe $\times 16$ (16 GB/s), benchmarks show that there are slight differences in performance. For instance, for a host-to-GPU0 scatter, host-GPU0 communication on DGX-1 consistently yields about 4% higher bandwidth (11.1 vs. 10.7 GB/s). Again, this overhead could be exposed for smaller batch sizes.

We next sketch a simple model of our performance expectations. With equivalent GPU performance, each workload has an identical GPU work cost. The expected overall DGX-1 performance advantage is therefore the workload's fraction of communication multiplied by the DGX-1's allreduce performance advantage. Given the DGX-1's slightly better allreduce performance on 4 GPUs for ResNet/x payloads, our model points to better DGX-1 performance for 2 and 4 GPUs. That expectation holds true in general. Given the GX8's better allreduce performance on 8 GPUs, the model suggests the 'knee' that appears in the DGX-1 curves for batch size 16. Recall that on 8 GPUs, the GX8 averages 10% better allreduce performance for the payloads used in ResNet/x (0.5 – 5 MB); see §4.2.

Our model does a good job explaining scaling results through 4 GPUs and the DGX-1 'knee' for batch 16. However, for batch sizes 32 and 64 on 8 GPUs, the DGX-1 consistently outperforms the GX8. Clearly, other effects must be taken into account to fully explain the scaling results.

We conclude by observing that similar performance trends hold true for a very large range of workload intensities, or activations/parameter per GPU (Fig. 7). These data show that if one is interested in ResNet-style workloads, the GX8 may be an attractive option if there is enough price differential.

5 Related Work

An important aspect of this work is defining ResNet/x, a paramerized version of ResNet. To our knowledge, there is no prior study that systematically parameterizes a deep learning workload to explore its space of computational intensities. Other performance evaluations select specific networks that have been designed for classifier performance, not workload evaluation. Even deep learning benchmark suites such as Fathom [22] represent only several points instead of a (discrete) continuum. Conversely, several studies assert general benefits [2, 23] of NVLink but do not look at the conditions under which one should or should not expect benefits.

Multi-GPU systems (or nodes) are becoming increasingly important. We have found no study comparing NVLink and PCIe-based GPU interconnects for up to 8 GPUs. Our comparison of the DGX-1/NVLink and GX8/Cirrascale SR3615 is relevant because both systems represent the 'top-tier' of multi-GPU systems but are also generally available.

Shams et al. [24] compare performance of Caffe using AlexNet for up to 4 P100 GPUs with and without NVLink. They show unexpected differences in the performance of AlexNet even with the use of only one GPU. Also, that study does not explain the effect of NVLink and the network topology using microbenchmarks.

Nomura et al. [25] study performance of a multi-GPU system connected by PCIe. They show significant speedup on 4 GPUs for applications of particle motion and advection computation. They showed that data transfer becomes a bottleneck even with relatively low computation.

Ben-Nun et al. [26] present the Groute asynchronous multi-GPU programming model and show nearly 7× speedup for some algorithms on a 8-GPU heterogeneous system. Awan et al.present MVAPICH2-GDR [27], an inter/intra-node multi-GPU collective library. They compare NVIDIA NCCL and MVAPICH2-GDR using microbenchmarks and a DNN training application. Their proposed design of MVAPICH2-GDR showed to have enabled up to 14× to 16.6× improvements as compared to NCCL-based solutions, for intra-/inter-node multi-GPU communication.

6 Conclusions

Scaling ML workloads across multiple on-node GPUs is becoming increasingly important. A closely related question is whether PCIe-based interconnects are adequate. We have provided a detailed performance evaluation of two GPU-based interconnects for eight NVIDIA Pascal P100 GPUs: (a) NVIDIA DGX-1 (NVLink 1.0 with 'hybrid cube mesh' topology); and (b) Cirrascale GX8 (two-level PCIe tree using two Cirrascale SR3615 switch risers). To systematically study the scaling effects of different neural networks, we define a parameterized variant of the popular ResNet [7] with controllable model activations and parameters.

To characterize the workload space, we defined a workload intensity metric that captures the expected computation/communication ratio and has good explanatory power. We show that our parameterized ResNet captures a large space of workload intensities.

Our conclusions are as follows. We find that the DGX-1 typically has superior performance. Given that the DGX-1's NVLink interconnect has more links and higher per-link bandwidth than the GX8's PCIe bus, this is not surprising. However, we also find that the GX8 is very competitive for all ResNet-style workloads. In rare cases, the GX8 slightly outperforms. The reason is related to the fact that the GX8's PCIe bandwidth saturates more quickly with respect to payload size. As a result, for medium-sized messages, the GX8 on 8 GPUs can have better memory copy latency and an average of 10% better allreduce performance. Our results shows that if one is interested in ResNet-style workloads, the GX8 may be an attractive option if there is enough price differential.

Acknowledgments

The authors thank Matthew Macduff (PNNL) for evaluation assistance. We are grateful for funding support from the U.S. Department of Energy's (DOE) Office of Advanced Scientific Computing Research as part of the "Center for Advanced Technology Evaluation" (CENATE) and "Convergence of Deep Learning and Machine Learning for HPC Simulation and Modeling." Pacific Northwest National Laboratory is operated by Battelle for the DOE under Contract DE-AC05-76RL01830.

References

1. Gawande, N.A., Landwehr, J.B., Daily, J.A., Tallent, N.R., Vishnu, A., Kerbyson, D.J.: Scaling deep learning workloads: NVIDIA DGX-1/Pascal and Intel Knights Landing. In: 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). (May 2017) 399–408
2. Foley, D., Danskin, J.: Ultra-performance Pascal GPU and NVLink interconnect. IEEE Micro **37**(2) (Mar 2017) 7–17
3. Sodani, A., Gramunt, R., Corbal, J., Kim, H.S., Vinod, K., Chinthamani, S., Hutsell, S., Agarwal, R., Liu, Y.C.: Knights Landing: Second-generation Intel Xeon Phi product. IEEE Micro **36**(2) (Mar 2016) 34–46
4. Cirrascale: The GX8 series multi-device peering platform. `http://www.cirrascale.com/documents/datasheets/Cirrascale_GX8Series_Datasheet_CM080C.pdf` (June 2016)
5. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., Weinberger, K., eds.: Advances in Neural Information Processing Systems 25. Curran Associates, Inc. (2012) 1097–1105
6. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 1–9
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. of the IEEE conference on Computer Vision and Pattern Recognition (CVPR). (2016) 770–778
8. Cirrascale: Cirrascale SR3615 PCIe switch riser (July 2015)
9. Cirrascale: Scaling GPU compute performance. `http://www.cirrascale.com/documents/whitepapers/Cirrascale_ScalingGPUCompute_WP_M987_REVA.pdf` (2015)
10. Luehr, N.: Fast multi-GPU collectives with NCCL. `https://devblogs.nvidia.com/parallelforall/fast-multi-gpu-collectives-nccl/` (April 2016)
11. NVIDIA: NCCL: NVIDIA collective communications library. `https://developer.nvidia.com/nccl` (August 2017)
12. Awan, A.A., Hamidouche, K., Venkatesh, A., Panda, D.K.: Efficient large message broadcast using NCCL and CUDA-aware MPI for deep learning. In: Proc. of the 23rd European MPI Users' Group Meeting. EuroMPI 2016, New York, NY, USA, ACM (2016) 15–22

13. Jeaugey, S.: Optimized inter-GPU collective operations with NCCL. `http://on-demand-gtc.gputechconf.com/gtc-quicklink/8Bdyh` (May 2017)
14. Patarasuk, P., Yuan, X.: Bandwidth optimal all-reduce algorithms for clusters of workstations. Journal of Parallel and Distributed Computing **69**(2) (2009) 117–124
15. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115**(3) (2015) 211–252
16. Berkeley Vision and Learning Center: Berkeley vision and learning center: Caffe. `http://caffe.berkeleyvision.org` (2016)
17. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
18. Berkeley Vision and Learning Center: Convolutional architecture for fast feature embedding (Caffe). `https://github.com/BVLC/caffe/` (2016)
19. NVIDIA: Convolutional architecture for fast feature embedding (Caffe). `https://github.com/NVIDIA/caffe` (2017)
20. Ben-Nun, T.: MGBench: Multi-GPU computing benchmark suite. `https://github.com/tbennun/mgbench` (Feb 2016)
21. Cirrascale: Cirrascale SR3514: Unexpected performance inequality. Technical Brief M901A - 092014
22. Adolf, R., Rama, S., Reagen, B., y. Wei, G., Brooks, D.: Fathom: reference workloads for modern deep learning methods. In: 2016 IEEE International Symposium on Workload Characterization (IISWC). (Sept 2016) 1–10
23. Christensen, C., Fogal, T., Luehr, N., Woolley, C.: Topology-aware image compositing using nvlink. In: 2016 IEEE 6th Symposium on Large Data Analysis and Visualization (LDAV). (Oct 2016) 93–94
24. Shams, S., Platania, R., Lee, K., Park, S.J.: Evaluation of deep learning frameworks over different HPC architectures. In: Proc. of the IEEE 37th Intl. Conf. on Distributed Computing Systems. (June 2017) 1389–1396
25. Nomura, S., Mitsuishi, T., Suzuki, J., Hayashi, Y., Kan, M., Amano, H.: Performance analysis of the multi-gpu system with expether. SIGARCH Comput. Archit. News **42**(4) (December 2014) 9–14
26. Ben-Nun, T., Sutton, M., Pai, S., Pingali, K.: Groute: An asynchronous multi-gpu programming model for irregular computations. In: Proc. of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, New York, NY, USA, ACM (2017) 235–248
27. Awan, A.A., Chu, C.H., Subramoni, H., Panda, D.K.: Optimized broadcast for deep learning workloads on dense-gpu infiniband clusters: MPI or NCCL? arXiv preprint arXiv:1707.09414 (2017)