A APPENDIX

A.1 Micro Benchmarks (DGX-1V)

We continue our discussion of micro benchmarks from Section 2.2, highlighting results for forwarding on a chain and fan in/out tests.

A.1.1 Depth Test

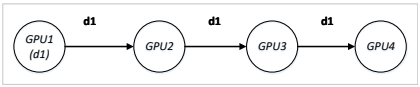
For the forwarding benchmark (Figure 20(a)), GPU1 is the source node with data named d1, and it passes the data d1 to GPU2 and then GPU2 forwards it to GPU3 etc. For "reduce+broadcast" (Figure 20(c)), we perform "reduce+forward" in one direction and "forward" in the other direction, as such a capability can be used for all-to-all reductions.

A.1.2 Breadth Test

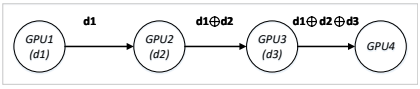
As illustrated in Figure 22(a), in fan-in forward, a center node (i.e. GPU4) collects data from multiple nodes and then forwards the collected data to its successor. Instead of just forwarding data, in the case of fan-in reduce+forward (Figure 22(b)), the center node computes a reduction function over the incoming data and its own data, then forwards the result to it successor. Fan-out forward (Figure 22(c)), is just the reverse of fan-in forward, in which the center node receives data from one node (i.e. GPU5), then multicasts the received data to its successors (i.e. GPU 1,2,3).

We experiment with different data size as we vary the number of GPUs that serve as fan-in source nodes or fan-out destination nodes. For DGX-1s, the maximum fan-in and fan-out degrees are limited to three. For brevity, we omit the graphs and highlight the key findings. Similar to the depth tests, with data size >50MB, fan-in and fan-out forward achieves near maximum throughput. Compared with fan-in forward, the throughput of fan-in reduce+forward decreases 1-2 GB/s on average due to the latency of launching reduction function kernels on the center node (GPU4).

Figure 22 depicts result of breadth tests with different data size as we vary the number of GPUs that serve as fan-in source nodes or fan-out destination nodes. We'd like to note that for the given topology of V100, the maximum fan-in and fan-out degrees are limited to three. In Figure 22(a), with data size >50MB, in all three cases, fan-in forward achieves near maximum throughput. Compared with fan-in forward, the throughput of fan-in reduce+forward (in Figure 22(b)) decreases 1-2 GB/s on average due to the latency of launching reduction function kernels on the center node (GPU4). We also note that running with 1000MB and a fan-in of 3 requires allocating memory for each incoming link and this exceeds the amount of memory available. Finally, for fan-out forward in Figure 22(c), the throughput is again close to the peak link bandwidth.



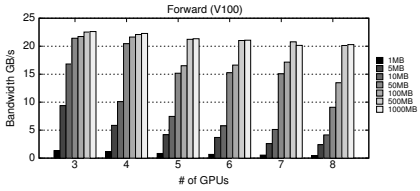
(a) chain forward



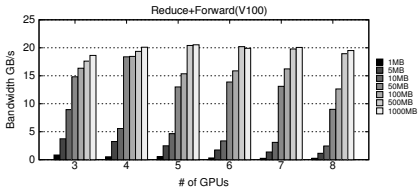
(b) chain reduce+forward

(c) chain reduce-broadcast

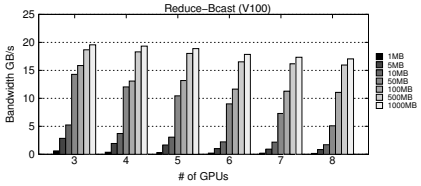
Figure 20. Depth test over a chain of GPUs.



(a) chain forward throughput



(b) chain reduce+forward throughput

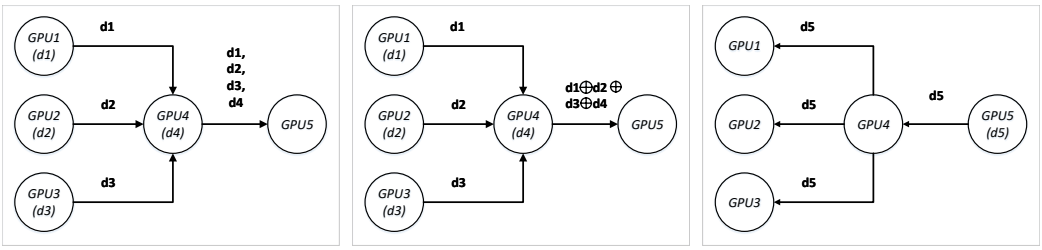


(c) chain reduce-broadcast throughput

Figure 21. Depth test throughput over a chain of GPUs.

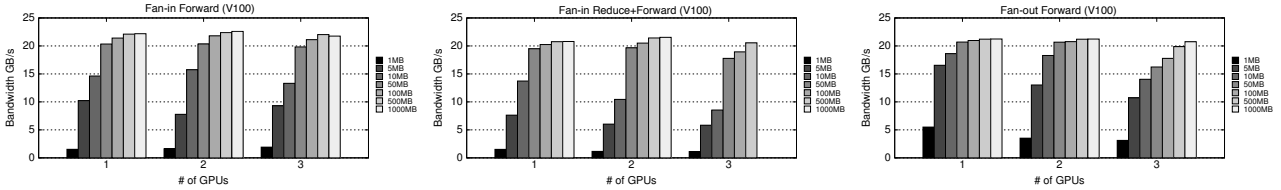
A.2 Exploiting Link Heterogeneity

For intra-node communication, servers such as the DGX-1 have both inter-GPU point-to-point (P2P) interconnects such as NVLink and shared interconnects such as PCIe (8-12GB/s) (PCI Express). PCIe connects multiple GPUs to each other within a machine, and to the CPU and IO deices, through a PCIe switch hierarchy. For inter-node communication, servers are equipped with multiple Ethernet or InfiniBand ports with a throughput of 3GB/s and 7GB/s per-port respectively. State-of-the-art collectives, such as NCCL and Horovod, all use ring-based protocols which fail to leverage link heterogeneity. The throughput of a ring is limited by the link with lowest bandwidth and hence these protocols either restrict themselves to high bandwidth,



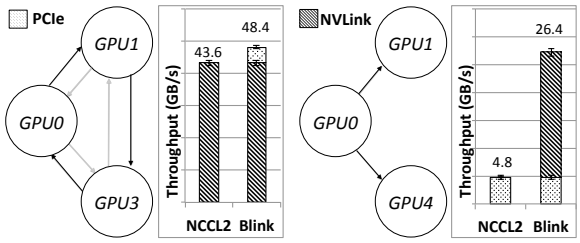
(a) Fan-in forward (b) Fan-in reduce+forward (c) Fan-out forward

Figure 22. Breadth test of data forward, reduce+forward in fan-in and fan-out topologies.



(a) Fan-in forward throughput (b) Fan-in reduce+forward throughput (c) Fan-out forward throughput

Figure 23. Breadth test throughput for Fan-in forward, Fan-in reduce+forward, Fan-out forward.



(a) Fully connected (b) Partially connected

Figure 24. Broadcast throughput, from GPU 0, using both NCCL and Blink on a DGX-1V.

homogeneous links, or limit throughput to the link with lowest bandwidth in the ring. For example, for multi-GPU communication within a machine, NCCL prioritizes using only NVLink over PCIe, as PCIe will be the bottleneck if included in a NVLink ring. Figure 24 shows an example 3 GPU setup for a `Broadcast` from GPU 0: when fully connected with NVLink, NCCL builds two rings (0->1->3->0 & 0->3->1->0) using bi-directional NVLinks, and ignores PCIe. If we replace GPU3 with GPU4, the lack of NVLink between GPUs 1 and 4 prevents NCCL from constructing NVLink-only rings and it has to fall back on PCIe based communication.

To handle heterogeneous links, `Blink` simultaneously transfers data on PCIe and NVLink within a machine and and balances the amount of data transferred across hybrid links. We next discuss how we handle hybrid PCIe and NVLink topologies in the context of our design presented above. The main challenge in using both PCIe and NVLink comes from the fact that NVIDIA driver does not directly allow users to control access to both links and if NVLinks are detected, the system will automatically enable P2P data transfer among GPUs using NVLinks. In our experience we find that using `cudaDeviceDisablePeerAccess` disables NVLinks and forces data transfer through PCIe links. However this still has the limitation that we cannot construct a unified topology with both sets of links. We address this problem by constructing two separate sets of trees, one over PCIe links and another over NVLinks.

One of the challenges with this approach is to balance the amount of data that is transferred over each link type. Our approach here is to minimize the maximum time taken by each of the transfers i.e. minimize $max(T_{PCIe}, T_{NVL})$.

We denote D_{total} as the total data needs to be transferred, and D_{PCIe}, D_{NVL} as the data size assigned on either PCIe or NVLink respectively. T_{dpa} is the latency for calling the `disable_peer_access()` and we denote BW_{PCIe} and BW_{NVL} as the bandwidth of PCIe and NVLink trees. Given this notation and objective, we can see that the optimal data split can be achieved by making $T_{PCIe} = T_{NVL}$.

$$\begin{aligned} &\text{Objective } T_{PCIe} + T_{dpa} = T_{NVL} \\ \implies D_{PCIe} =& \frac{D_{total} \times BW_{PCIe}}{BW_{PCIe} + BW_{NVL}} - \\ & \frac{T_{dpa} \times BW_{PCIe} \times BW_{NVL}}{BW_{PCIe} + BW_{NVL}} \\ D_{NVL} =& D_{total} - D_{PCIe} \end{aligned} \tag{8}$$

The optimal data splits are shown in Equation 8. Note that in Equation 8, T_{dpa} is empirically measured and may vary depending on number of GPUs. We measure this during the initial few calls into our library.

We evaluate hybrid (or combined) data transfers over both

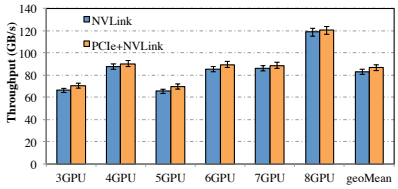


Figure 25. Hybrid and NVLink-only broadcast throughput comparison with varied number of GPUs.

PCIe *and* NVLink. For brevity, we only show broadcast results for 3-8 GPUs on the AWS DGX-1V server. Figure 25, highlights the additional 2-5 GB/s performance gain over NVLink-only transfers when `Blink` combines transfers over both NVLink and PCIe. The time to switch communication channels from NVLink to PCIe increases as the number of GPUs grow. For 3 and 4 GPU settings, compared with NVLink-only Broadcast, hybrid transfers can achieve around 5GB/s boost; with 7 and 8 GPUs this boost is only around 2GB/s. This is because the total time spent on enabling and disabling peer-access, i.e. switching between PCIe and NVLink, is proportional to the number of GPU in use.

A.3 DGX-2 Allreduce

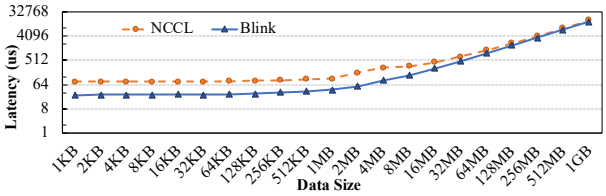


Figure 26. Allreduce Latency in μs (`Blink` and NCCL2) on a 16-GPU DGX-2.

We present above results comparing latency for AllReduce operations when using 16 GPUs on a DGX-2 machine. As described in Section 3.4, Blink uses a number of single-hop trees to perform AllReduce when GPUs are connected using NVSwitch. One of the main advantages of a single-hop tree is that this reduces latency compared to using a ring across the GPUs. To validate this we measure the latency of AllReduce and vary the dataset size from 1KB to 1GB as shown in Figure 26. We find that Blink is especially effective for smaller data sizes offering up to 3.32$\times$ lower latency compared to NCCL's double-binary trees and rings.