Speeding up Collective Communications Through Inter-GPU Re-routing

Kiran Ranganath, AmirAli Abdolrashidi, Shuaiwen Leon Song, and Daniel Wong, *Member, IEEE*

Abstract—In order to address the vast needs of disparate domains, computing engines are becoming more sophisticated and complex. A typical high-performance computational engine is composed of several accelerator units, in most cases GPUs, plus one or more CPU controllers. All these components are becoming increasingly interconnected to satisfy bandwidth and latency tolerance demands from modern workloads. Due to these constraints, solutions to efficiently interconnect them or to systematically manage their traffic—such as PCIe v3, NVLink v1 and v2 on the hardware side, and NVIDIA Collective Communication Library (NCCL) and AMD ROCM layer on the software side—are becoming more commonplace inside HPC systems and cloud data centers. However, as the number of accelerators increases, workloads (especially machine learning) might not be able to fully exploit the computational substrate due to inefficient use of hardware interconnects. Such scenarios can lead to performance bottlenecks where high-bandwidth links are not used by the underlying libraries and under-performing links are overused.

This work proposes Workload Optimization Through Inter-GPU Re-routing (WOTIR), which consists of enhanced NCCL-based collective primitives that aim to boost bandwidth utilization (through more efficient routing) and reduce communication overhead. WOTIR targets GPUs with no direct NVLink communication path (which leads to PCIe communications) and instead re-routes communication through intermediate GPUs to bridge NVLink segments and avoid PCIe communications. Such method allows the maximum possible utilization of the NVLink bandwidth between the GPUs without routing through the PCIe bus. Using this method, we see a reduction of up to 34% in execution time for selected machine learning workloads when non-optimal GPU allocations arise.

Index Terms—Collective communication, GPU, Interconnect

1 Introduction

Accelerator-based computing has become mainstream for many domains that have high computational needs. The most common accelerator architecture used in current systems is GPU-based. The domains that best exploits this increased computational power are usually HPC applications which are highly concurrent and floating point intensive, and Machine Learning (ML) workloads that require a large number of small computations and efficient communication. Moreover, these workloads continue to evolve to efficiently map to the ever-changing computational substrate.

Such computational pipelines have been augmented to take advantage of optimized collective operations provided for the hardware. Vendor-provided solutions such as the *NVIDIA Collective Communications Library (NCCL)* [1] exploit the system under "normal" conditions. However, due to complexities that arise from advanced workflows, frameworks, and multi-user environments, these solutions fail to fully leverage the underlying compute substrate. Domains like machine learning training, in which the process can take days, and graph-based analytics in cloud environments [2], [3] can suffer from these pathological scenarios, since several workloads might be required to co-exist inside a single compute node. Such partitioning of resources might lead to "bad" placements in which GPUs might become isolated[1] for a single workload, leading to inefficient communication.

1.1 Multi-GPU Server Architectures

The NVIDIA DGX-1 system was one of the first multi-GPU reference server design released. This design has been adopted by various supercomputers and cloud data centers. Examples include the Summit Supercomputer [4], Microsoft Olympus [5], and Facebook Big Basin [6].

Figure 1 illustrates the GPU network topology in the DGX-1 system. In this server, two CPUs manage 4 GPUs through a PCIe link each (i.e. 8 GPUs in total). Every GPU has direct peer-to-peer access to 3 other GPUs through single and double NVLink connections (NVIDIA's high speed GPU-to-GPU interconnect [7]) as shown. In the DGX-1 systems (that use NVIDIA P100 GPUs), NVLink-v1 can reach up to 20 GB/s per link. Newer systems using V100s GPUs employ NVLink-v2 which can achieve up to 25 GB/s per link with the same network topology. In these configurations, the black arrows in Figure 1 signify peer-to-peer connectivity. Two black arrows mean double NVLink connections. The CUDA runtime API enables NVLink-based device-to-device communication only if the pair of GPUs are directly connected via NVLink [8].

1.2 Motivation - The Fragmentation Problem

To highlight potential resource allocation and communication issues in cloud environments that can arise in multi-GPU servers, let us consider the following scenario. There is a job queue, where queued Jobs 1 through 4 utilize 1, 4, 2 and 2 GPUs respectively. These jobs can either be batch-type or latency-critical that are provisioned for request-response type workloads. Let us assume a naive scheduling scenario[2] where Job 1 takes GPU 0; Job 2 takes GPUs 1, 2, 3 & 4; and Job 3 takes GPUs 5 & 6. Since there is only one GPU left, and Job 4 needs 2 GPUs, it waits until 2 GPUs are available. Now assume Job 1 frees GPU 0, and Job 4 is scheduled on GPUs 0 & 7. In the GPU topology shown in Figure 2a, GPUs 0 & 7 do not have NVLink connectivity. Although it is desirable to place workloads on adjacent GPUs for faster communication, some of them may already be occupied. Therefore, we would be forced to place the workloads to non-adjacent GPUs, which can add to the communication overhead. We call this problem *fragmentation*. Hence Job 4 running on GPUs 0 and 7 suffers from this communication overhead.

- *K. Ranganath, A. Abdolrashidi, and D. Wong are with the University of California Riverside, Riverside, CA 92521. E-mail: {kiran.ranganath, amirali.abdolrashidi}@email.ucr.edu, daniel.wong@ucr.edu.*
- *S.L Song formarly with PNNL is currently with University of Sydney, Camperdown NSW 2006, Australia. Email: leonangel991@gmail.com.*

Manuscript submitted: 29-May-2019. Manuscript accepted: 26-Jun-2019. Final manuscript received: 3-Jul-2019.

1. without a direct high-bandwidth communication link between them

2. which is the current standard policy in NVIDIA Docker to allocate GPUs in cloud environments

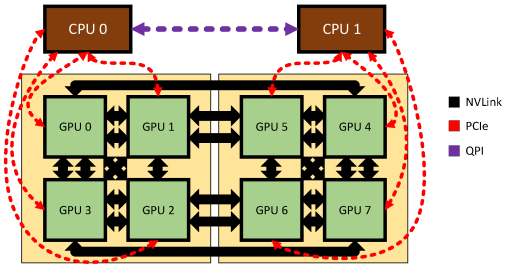
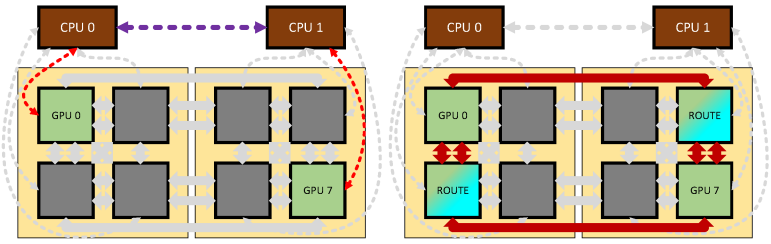


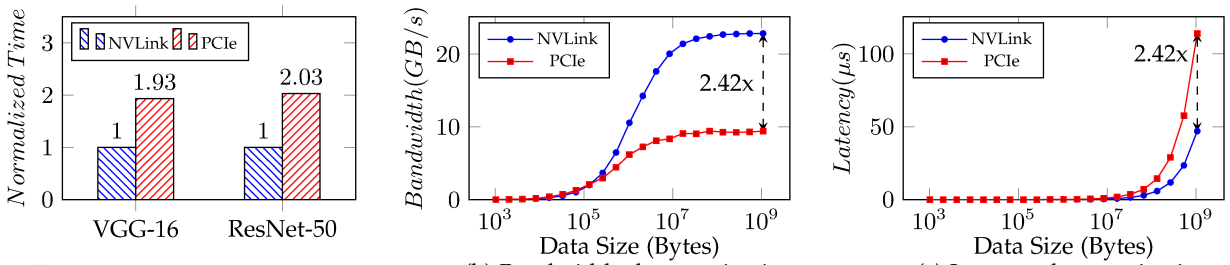
Fig. 1: NVIDIA DGX-1 topology



(a) GPUs 0 and 7 use PCIe since there is no direct NVLink

(b) WOTIR establishes NVLink route using *route-GPUs*.

Fig. 2: Placement scenario



(a) Caffe Training time normalized to NVLink (b) Bandwidth characterization (c) Latency characterization

Fig. 3: Placement effects in GPU-to-GPU communication

1.2.1 *Characterizing effect of poor placement*

The effect on the execution time of a non-optimal placement can be seen in Figure 3(a). This figure shows execution times normalized to NVLink, i.e. an optimal placement using GPUs 0 and 1. We observe the execution time using PCIe (GPUs 0 and 7) to be as much as 2x compared to NVLink. To further demonstrate the performance gap between PCIe and NVLink, we present the bandwidth and latency of the memory copy operation in Figures 3(b) and 3(c). The memory copy is performed on contiguous data allocations with sizes ranging from 1 KB to 1 GB. The bandwidth of the memory copy operation saturates NVLink at a little over 22 GB/s and PCIe at 9 GB/s.

2 WOTIR

In order to alleviate the degraded network performance due to poor workload placement within a single node, we propose Workload Optimization Through Inter-GPU Rerouting (WOTIR). In this technique, we augment NCCL to use two-hop NVLink-based routing to circumvent PCIe bottlenecks, thereby improving the overall performance. This work courrently focuses on tolerating GPU fragmentation within a server node. We shall leave the study of cluster-level GPU fragmentation for future work.

2.1 NCCL

NCCL implements various collective communication patterns using coordinated primitives. These primitives operate on data in the send/receive buffers of the GPU, as well as data in the GPU's memory. The following primitives are used in NCCL— *Copy*: Copy the data in receive buffer to GPU's memory or from GPU's memory to send buffer. *Double Copy*: Copy the data in receive buffer to GPU's memory and send another copy to the next GPU by copying on to send buffer. *Reduce*: Perform an arithmetic or logical reduce operation on data in the receive buffer and a location in GPU memory, and copy the final result in another location in the GPU memory. *Reduce-Copy*: Perform arithmetic or logical reduce operation on data in receive buffer and location in GPU memory and place one copy at a location in its GPU memory and place the second copy onto the send buffer so the data is available to the next GPU.

To facilitate communication, NCCL organizes participating GPUs into *rings*. During the setup of these rings, communication channels (PCIe or NVLink) are enabled as shown in Figure 2b. These channels assist data movement among participating GPUs through primitive operations.

2.2 NVLink Routing

In WOTIR, we implemented a *route-GPU* to transport data without using PCIe. During initialization, we perform a static lookup of the topology matrix to determine the ideal route for GPUs without a direct NVLink connection. Between any two non-NVLink connected GPUs, there exist two possible router GPUs. For example, between GPU 2 and 7, possible routing GPU candidates are 3 and 6. In our current implementation, we naively select the GPU that would complete a NVLink-only ring.

After identifying a GPU as a *route-GPU*, we launch a route kernel on a single warp to facilitate the data movement. This kernel performs a new primitive called *forward*. In this primitive, data is directly transferred from the receive buffer to the send buffer. Unlike existing NCCL primitives, the forward primitive does not require the route GPU to have data buffers in the GPU's memory, only the send/receive buffer of the NCCL communication rings. In order to synchronize the coordination of data transfers in the collective operation, the routing GPU will also have to insert itself into the flagging mechanism that currently exist in NCCL rings.

The route kernel persists until the job is complete. Since WOTIR maintains NCCL's APIs, no change is required by the application to benefit from NVLink routing.

To illustrate the functionality of WOTIR we present the following illustration. For example, if GPUs 0 & 7 were assigned for the application as depicted in Figure 2a, the communication between 0 & 7 would take place via PCIe bus due to a lack of NVLink connection between them. However, with WOTIR, GPUs 3 & 4 are assigned as *route-GPUs* as shown in Figure 2b, and data is forwarded through NVLink, eliminating the need for PCIe.

3 Evaluation

We perform our evaluation on NVIDIA DGX-1 V-100 system with 8 V-100 GPUs with global memory of 16 GB each. We evaluate our work at two levels: memory copy performance, and execution time improvement.

3.1 Memory Copy Performance

To demonstrate the benefit of WOTIR, we repeat the memory copy experiment leveraging our framework, and present the results in Figure 4. This figure shows the latency and bandwidth transferring data between 2 GPUs using NVLink, PCIe, and WOTIR. The peak bandwidth achieved in WOTIR is nearly 18 GB/s which is 2x peak bandwidth achievable in the same combination of GPUs (0 & 7) with PCIe. WOTIR does however exhibits poor bandwidth utilization for smaller data transfer sizes due to NVLink overheads, a phenomenon showcased in [9]. This degradation could be avoided by extending WOTIR to employ an adaptive communication strategy to ensure performance to be at least equal to PCIe but not worse.

3.2 Machine Learning Workloads

One prominent use of multi-GPU environments is Machine Learning (ML). Several ML frameworks have been designed to leverage multiple GPUs including Caffe, TensorFlow, and PyTorch. We demonstrate WOTIR using Caffe, an open-source deep learning framework which can be accelerated with GPUs. We train 4 ML models—AlexNet [10], VGG-16 [11], ResNet-50 [12], and GoogleNet [13] since these models are some of the most heavily used in deep learning. We measure the execution time of training for *max_iterations=10,000* on GPU combinations with NVLink, PCIe, and WOTIR.

Figures 5 and 6 showcase execution times compared to PCIe and NVLink respectively. In Figure 5, we see that WOTIR reduced the execution time by 34% in VGG-16 compared to PCIe. It is also noteworthy that in AlexNet, WOTIR performs better than NVLink. While the device-to-device communication is large enough to utilize the full bandwidth enable by WOTIR, the majority of the communication is host-to-device. This is an artifact of AlexNet's inter-layer communication prevalent during training. By using two GPUs managed by two separate CPUs, we effectively parallelize the host-to-device communication.

In Figure 6, we observe an increase in execution time due to PCIe overheads. The increase is around 10% for AlexNet and GoogleNet, whereas in VGG-16 and ResNet-50, the PCIe overheads increase the execution times up to 120%. Hence we see that VGG-16 and Resnet-50 are the biggest beneficiaries of WOTIR in our evaluation.

3.3 Interference on route-GPU

To analyze the interference of the routing operation on workloads running on the router GPU, we use BabelStream [14] to stress GPU memory bandwidth while performing a multiple memory copies of 1 KB to 1 GB. We ran 1000 iterations of BabelStream to account for experimental variations and present the measured distribution in Table 1. Each timing represents one iteration of BabelStream which stresses the memory bandwidth by performing a copy operation of 16 GB to the global device memory. We observed that for the majority of the case, memory bandwidth interference on the route GPU is minimal. We only observe interference for a minority of copies at the tail. Further work to investigate how to reduce routing operation overhead is warranted. Overall, we observed negligible computational overhead because the routing operation only requires one warp out of the entire GPU.

TABLE 1: Distribution of BabelStream turnaround time observed during 1000 iterations of Interference Analysis

State	Min (ms)	25th% (ms)	Median (ms)	75th% (ms)	95th% (ms)	Max (ms)
w/o Routing	2.033	2.034	2.035	2.036	2.040	3.140
w/ Routing	2.034	2.035	2.036	2.053	5.674	11.459

4 Related Works

Multi-GPU clusters are increasingly being used for many large-scale applications at a rapid rate, including deep learning and high-performance computing [9], [15]. Recently there have been novel attempts to improve the workload placement and inter-GPU communication and network topology.

[16] propose Blink, a set of tree-based broadcast collective communication primitives for multiple GPUs. [17] have addressed the effect of the topology of GPU networks within a node on the performance, and propose a topology-aware GPU assignment to MPI tasks to improve the communication and performance. [18] present ADAPT, an Open MPI-based collective communication framework which uses a topology-aware communication tree to improve collective operations and maximize the number of communications across the systems at any time. [19] introduce Gandiva, a cluster-scheduling framework for deep learning across multiple GPU clusters. However, the problem of slower PCIe communication remains in these works, which can exacerbate the performance. Our paper addresses and resolves this issue using router GPUs to forward data among the devices.

Gandiva also proposed migration of jobs to GPUs that are freed while the job is running. Since the scheduler constantly checks on GPUs to be freed, the communication problem persists until the job is potentially migrated. Moreover, as mentioned in [20], [21], [22], preemption and migration of GPU kernels have a significant cost. Furthermore, migration policies require GPUs to be more dependent on CPUs to constantly monitor the availability of a better allocation for the running jobs. In contrast, our proposed technique has the potential to provide near-optimal communication performance without the need for migration or preemption.

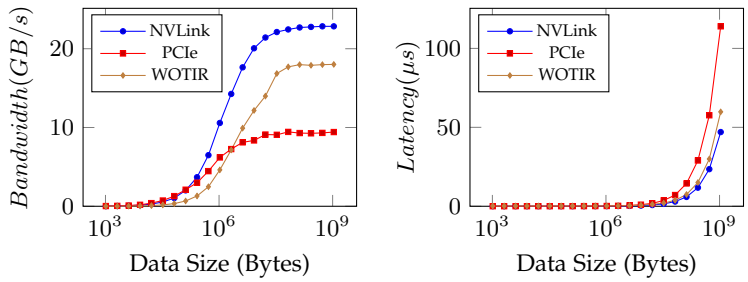


Fig. 4: Bandwidth and Latency improvement with WOTIR.

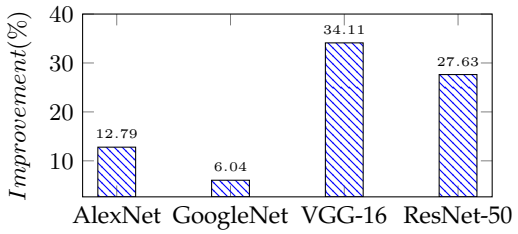


Fig. 5: Execution time improvement in Caffe of WOTIR compared to PCIe for 3 GPUs

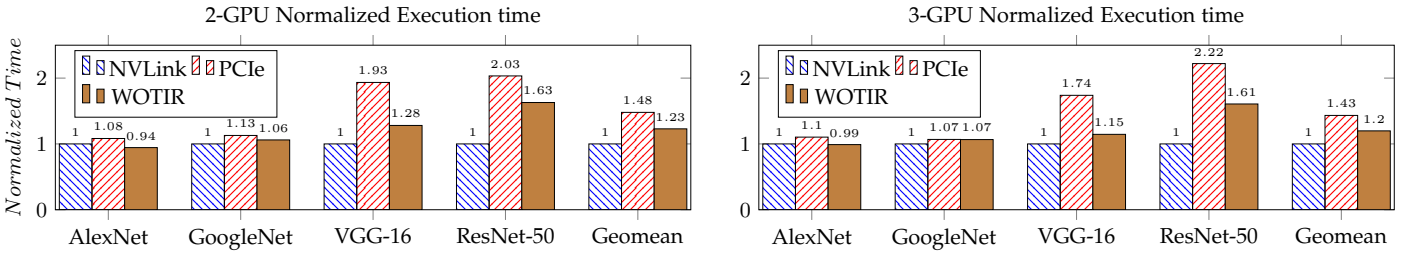


Fig. 6: Execution time of PCIe and WOTIR normalized to NVLink.

5 Conclusion

In this paper, we proposed WOTIR, a set of collective communication primitives designed to transfer data among the GPUs in a multi-GPU system through the use of router-GPUs. WOTIR uses NVLinks to forward the data traffic between communicating GPUs that would otherwise only communicate through the PCIe bus due to inefficient GPU mapping. We have tested this method on a variety of Caffe ML models, achieving execution time reduction of up to 34% compared to PCIe.

Acknowledgements

This research is supported by U.S. DOE Office of Science, Office of Advanced Scientific Computing Research, under the CENATE project (award No. 66150), The Pacific Northwest National Laboratory is operated by Battelle for the U.S. Department of Energy under contract DE-AC05-76RL01830. This work is also partially funded by NSF grant CCF-1815643 and the University of California, Riverside.

References

[1] "Nvidia collective communication library (nccl) documentation," https://docs.nvidia.com/deeplearning/sdk/nccl-developer-guide/docs/index.html, 2016, accessed 04-20-2019.

[2] Amazon, "Amazon ec2 elastic gpus." [Online]. Available: https://aws.amazon.com/ec2/elastic-gpus/

[3] Microsoft, "Microsoft azure nc virtual machines-gpu compute," November 2017. [Online]. Available: https://azure.microsoft.com/en-us/blog/azure-n-series-general-availability-on-december-1/

[4] J. Hines, "Stepping up to summit," *Computing in Science & Engineering*, vol. 20, no. 2, pp. 78–82, 2018.

[5] S. Tavallaei and R. Ober, "Microsoft project olympus hyperscale gpu accelerator(hgx-1)," Microsoft Azure, Tech. Rep., 2017.

[6] K. Hazelwood, S. Bird, D. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, M. Fawzy, B. Jia, Y. Jia, A. Kalro *et al.*, "Applied machine learning at facebook: A datacenter infrastructure perspective," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2018.

[7] D. Foley, "Nvlink, pascal and stacked memory: Feeding the appetite for big data," *Nvidia. com*, 2014.

[8] NVidia, "Tuning cuda applications for volta." [Online]. Available: https://docs.nvidia.com/cuda/volta-tuning-guide/index.html

[9] A. Li, S. L. Song, J. Chen, J. Li, X. Liu, N. Tallent, and K. Barker, "Evaluating modern gpu interconnect: Pcie, nvlink, nv-sli, nvswitch and gpudirect," *arXiv preprint arXiv:1903.04611*, 2019.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[12] 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.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[14] T. Deakin, J. Price, M. Martineau, and S. McIntosh-Smith, "Gpu-stream v2. 0: benchmarking the achievable memory bandwidth of many-core processors across diverse parallel programming models," in *International Conference on High Performance Computing*, 2016.

[15] A. A. Awan, C.-H. Chu, H. Subramoni, and D. K. Panda, "Optimized broadcast for deep learning workloads on dense-gpu infiniband clusters: Mpi or nccl?" in *Proceedings of the 25th European MPI Users' Group Meeting*, 2018.

[16] G. Wang, A. Phanishayee, S. Venkataraman, and I. Stoicat, "Blink: A fast nvlink-based collective communication library," 2018.

[17] I. Faraji, S. H. Mirsadeghi, and A. Afsahi, "Topology-aware gpu selection on multi-gpu nodes," in *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2016.

[18] X. Luo, W. Wu, G. Bosilca, T. Patinyasakdikul, L. Wang, and J. Dongarra, "Adapt: an event-based adaptive collective communication framework," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, 2018.

[19] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang *et al.*, "Gandiva: Introspective cluster scheduling for deep learning," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2018.

[20] J. J. K. Park, Y. Park, and S. Mahlke, "Chimera: Collaborative preemption for multitasking on a shared gpu," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2015.

[21] I. Tanasic, I. Gelado, J. Cabezas, A. Ramirez, N. Navarro, and M. Valero, "Enabling preemptive multiprogramming on gpus," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, 2014.

[22] S. Kato, K. Lakshmanan, A. Kumar, M. Kelkar, Y. Ishikawa, and R. Rajkumar, "Rgem: A responsive gpgpu execution model for runtime engines," in *2011 IEEE 32nd Real-Time Systems Symposium*, 2011.