Tartan: Evaluating Modern GPU Interconnect via a Multi-GPU Benchmark Suite

Ang Li[†], Shuaiwen Leon Song[†‡], Jieyang Chen[†§], Xu Liu[‡], Nathan Tallent[†], and Kevin Barker[†]
[†]Pacific Northwest National Laboratory, [‡]College of Williams and Mary, [§]University of California Riverside
{ang.li, shuaiwen.song, jieyang.chen, nathan.tallent, kevin.barker}@pnnl.gov, xl10@cs.wm.edu

***Abstract*—High performance multi-GPU computing becomes an inevitable trend due to the ever-increasing demand on computation capability in emerging domains such as deep learning, big data and planet-scale applications. However, the lack of deep understanding on how modern GPUs can be connected and the actual impact of state-of-the-art interconnect on multi-GPU application performance becomes a hurdle. Additionally, the absence of a practical multi-GPU benchmark suite poses further obstacles for conducting research in multi-GPU era. In this paper, we fill the gap by proposing a multi-GPU benchmark suite named Tartan, which contains microbenchmarks, scale-up and scale-out applications. We then apply Tartan to evaluate the four latest types of modern GPU interconnects, i.e., PCI-e, NVLink-V1, NVLink-V2 and InfiniBand with GPUDirect-RDMA from two recently released NVIDIA super AI platforms as well as ORNL's exascale prototype system. Based on empirical evaluation, we observe four new types of NUMA effects: three types are triggered by NVLink's topology, connectivity and routing, while one type is caused by PCI-e (i.e., anti-locality). They are very important for performance tuning in multi-GPU environment. Our evaluation results show that, unless the current CPU-GPU master-slave programming model can be replaced, it is difficult for scale-up multi-GPU applications to really benefit from faster intra-node interconnects such as NVLinks; while for inter-node scale-out applications, although interconnect is more crucial to the overall performance, GPUDirect-RDMA appears to be not always the optimal choice. The Tartan benchmark suite including the microbenchmarks are opensource and available at *http://github.com/uuudown/Tartan*.**

I. Introduction

Multi-GPU execution becomes an inevitable trend for warehouse GPGPU computing. This is due to the ever-increasing demand of computation capability from emerging domains such as machine learning, big data and planet-scale applications [1], [2]. With increasingly larger problem to solve, scalable GPU computing is required. Recently, a research group from *Facebook* leveraged 256 GPUs to train a *ResNet-50* neural network on *ImageNet* within one hour, achieving 90% GPU scaling efficiency [1]. The *Swiss National Supercomputing Center* (*CSCS*) relied on 4888 GPUs in the *Piz Daint* supercomputer to simulate near-global climate in ultra-high resolution [2].

Multi-GPU execution scales in two directions: *vertically scaling-up in a single node* and *horizontally scaling-out across multiple nodes*. A good example to describe the intra-node scale-up scenario is the recent NVIDIA DGX-1 AI server [3], which incorporates eight Pascal-P100 or Volta-V100 GPUs connected via fast NVLink interconnect. For the inter-node scale-out scenario, the U.S. *Department of Energy* (*DoE*) has announced the deployment of two GPU-accelerated supercomputers *Summit* and *Sierra* in *Oak Ridge* and *Livermore National Laboratories* this year, with more than 3400 GPU-integrated nodes interconnected by InfiniBand.

However, gaining performance from multi-GPU is not an easy task, mainly because (i) there is no standard multi-GPU parallel programming, execution and performance models, partially due to the limited knowledge on how modern GPUs are interconnected and their communication patterns; (ii) traditionally, inter-GPU communication shares the same bus as CPU-GPU communication, such as PCI-e bus. Recently, the situation seems to be changed due to the introduction of NVLink. However, the characteristics of NVLinks as well as their performance impact on real-world multi-GPU applications are still unknown; (iii) there is no common practical multi-GPU benchmark suite acting as the baseline for evaluating multi-GPU related techniques and conducting fair comparison.

In this paper, we fill this gap by thoroughly evaluating different types of modern GPU interconnects, including PCI-e, NVLink Version-1, NVLink Version-2 and InfiniBand with GPUDirect-RDMA on real HPC systems using the proposed Tartan multi-GPU benchmark suite. Comprising microbenchmarks, seven scale-up and seven scale-out multi-GPU applications, Tartan is designed to deeply characterize the employed interconnects, including startup latency, sustainable uni/bi-direction bandwidth, network topology, communication efficiency, routing and NUMA effects, under the two major communication patterns: *Peer-to-Peer(P2P)* and *Collective(CL)*.

Our evaluation on multi-GPU execution can benefit software designers by gaining deep knowledge about modern GPU interconnects. These characterization pave the way for building more mature multi-GPU programming, execution and performance models and reliable simulators for better guiding application development and performance tuning. Furthermore, Tartan aims to foster a community efforts for enabling future multi-GPU research. It offers a general baseline for software and hardware designers to assess their proposed techniques for multi-GPU execution; application developers may also find it useful on providing basic examples about how to port their codes towards modern multiGPU-accelerated HPC platforms. This paper thus makes the following contributions:

- We propose a multi-GPU benchmark suite called Tartan containing practical microbenchmarks, scale-up and scale-out applications for interconnect characterization, as well as intra- and inter-node performance evaluation.
- We use Tartan to characterize four modern GPU interconnects, including PCI-e, NVLink-V1, NVLink-V2 and IB with GPUDirect-RDMA from the state-of-the-art multi-GPU-accelerated HPC platforms. Based on the results, we summarize several observations, challenges to address, and potential research topics regarding to multi-GPU execution.

Our microbenchmarking leads to the following major findings:

- For intra-node GPU P2P communication, we address three types of NUMA effects due to specific position, connectivity and routing choice, respectively, for NVLink. We also find a new type of NUMA effect for PCI-e, labeled as *anti-locality*.

- For intra-node GPU collective communication, the latency increases linearly with more participant GPUs as expected. However, the bandwidth declines for PCIe while increases for NVLink due to distinct network topology (i.e., tree vs. hypercube). In addition, NVLink shows optimal overall bandwidth with 4 or 8 GPUs, but worst with 5 GPUs.
- For inter-node GPU P2P communication, GPUDirect-RDMA essentially exhibits inferior performance than copying through pinned system memory in our testing. The performance even degrades with larger data arrays.
- For inter-node GPU collective communication, both latency and bandwidth keep almost unchanged with more participant nodes (except *all_reduce*). Meanwhile, *all_gather* and *reduce_scatter* exhibit significantly lower than average bandwidth when there are 3 or 5 nodes in the communication.

II. Modern GPU Interconnect

We focus on discussing four types of modern interconnects used for inter-GPU communication: *PCI-e*, *NVLink-V1*, *NVLink-V2* and *InfiniBand*. Table I lists the HPC platforms we adopted for evaluation: *P100-DGX-1* and *V100-DGX-1* are for evaluating intra-node interconnect, including PCI-e, NVLink-V1 and NVLink-V2. ORNL's *SummitDev* is for assessing inter-node InfiniBand interconnect with GPUDirect-RDMA.

A. Intra-Node Interconnect

PCI-e: *Peripheral-Component-Interconnect-Express-Bus*, or PCI-e, is a high-speed serial computer expansion bus standard. Traditionally, a GPU-integrated system connect one or multiple GPU devices to the CPUs via PCI-e. However, compared to the interconnect between CPU and DRAM, PCI-e is much slower. It often becomes a major performance bottleneck for GPU-acceleration [4], [5], [6]. Such a condition exacerbates when GPU P2P communication is enabled [3].

The dash-lines in Figure 1-(A) illustrate how the eight GPUs are interconnected by PCI-e and QPI in DGX-1. As is shown, the PCI-e network in DGX-1 forms a balanced tree structure, e.g., GPU-0 and GPU-1 are connected via a PCI-e switch. The switch is further connected to CPU socket-0. A similar scenario applies to other GPUs. Finally, the dual CPU sockets are bridged by *QuickPath Interconnect*, or QPI.

NVLink-V1: NVLink is a novel wire-based communication interface for near-range devices based on *High-Speed-Signaling-Interconnect* (NVHS) [3], [7]. It supports P2P communication that enables CPU-GPU or GPU-GPU linking. It allows direct read and write on remote CPU's main-memory and/or peer GPU's device-memory. Remote atomic operation is also feasible. NVLink is bidirectional; each link consists of two sublinks — one for each direction. Each sublink further contains eight differential NVHS lanes. An embedded clock is integrated for transmission. The packet size varies from 16 bytes to 256 bytes (one 128-bit flit to 16 128-bit flits). The communication efficiency is strongly correlated to packet size. Overall, it is reported to be twice as efficient as PCI-e [7].

A NVLink can be viewed as a cable with two terminal-plugs whereas each GPU incorporates several NVLink slots. How these slots are connected via the NVLink cables dictate the topology and bandwidth of the GPU network. Multiple cables can be ganged together to enhance bandwidth when they are linking the same endpoints. A Pascal-P100 GPU has quad NVLink slots. Therefore, for a dual-GPU system, a direct setting would be two GPUs connected by four NVLinks, leading to 4× bandwidth of a single link.

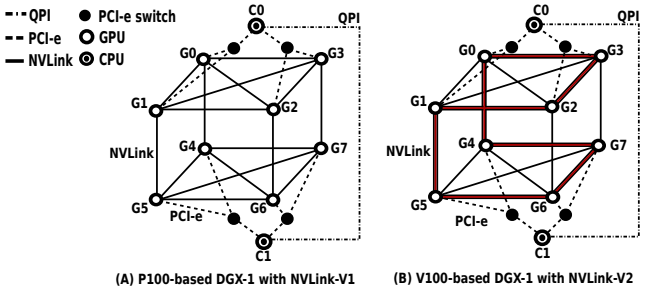

Fig. 1: PCI-e and NVLink interconnect topology for DGX-1 machines

The GPU network topology for DGX-1 is known as *Hypercube Mesh*. As shown in Figure 1-(A), each GPU occupies a corner of the cube and all the 12 edges are NVLink connections. For the upper and lower layers, the diagonals are also connected, forming two fully-connected subsets. Such a topology design is balanced, with stronger connectivity inside a layer. In other words, accessing within a layer is UMA, while accessing nodes across layers leads to NUMA effect (when they are not directly linked, e.g., from GPU-0 to GPU-7). In fact, NVLink is not self-routed when the two terminals are not directly linked. It relies on explicit routing through a user-specified intermediate node (further discuss in Section III-A).

NVLink-V2: The second generation of NVLink improves per-link bandwidth and adds more link-slots per GPU: in addition to 4 link-slots in P100, each V100 GPU features 6 NVLink slots; the bandwidth of each link is also enhanced by 25%. In addition, a low-power operating mode is introduced for saving power when a link is not being heavily exploited.

The extra two link-slots have enabled certain alternation to the original network topology. As shown in Figure 1-(B), the V100-DGX-1 does not choose to further strengthen the connectivity within a layer, but forming a fast *Backbone Ring* inside the Hypercube Mesh network. Each connection in this ring enables 2× bandwidth of a normal link. We suspect this is to improve the efficiency of NCCL communication library, as further discussed in Section III-B.

B. Inter-Node Interconnect with GPUDirect-RDMA

InfiniBand: We will not discuss InfiniBand [8] itself since it is already widely used for HPC today and has been extensively studied. Instead, we focus on its relation with GPU. Since the Kepler architecture, NVIDIA GPUs[1] have introduced *GPUDirect-RDMA* [10] (Correspondingly, AMD proposed *ROCn-RDMA* [11]). It enables third-party PCI-e devices, especially the IB *Host-Channel-Adapter* (i.e., HCA) to directly access GPU device memory via PCI-e without any assistance from CPU or staging through the main memory, which significantly improves the efficiency of inter-node GPU communication. To achieve IB RDMA, GPU vendors offer an OS kernel extension to return a DMA bus mapping for GPU device memory. When a user creates an IB region, it signals the IB driver with the target address of the GPU device memory. IB driver then calls a routine to obtain the DMA mapping. Finally, a normal IB virtual memory structure is returned to the user program as if it targets normal CPU memory.

[1]In this paper, we use CUDA [9] terminology for description and evaluation. However, the evaluation techniques and the Tartan benchmark suite can be applied to other GPUs as well.

TABLE I: Evaluation Platforms. Mem refers to memory. MBand refers to memory bandwidth. CC refer to compiler. Arch refers to GPU architecture generation. SP/DP Flops refer to GPU theoretical single/double-floating-point performance. MB/W refer to GPU device memory bandwidth. Rtm. refers to CUDA runtime.

Platform	Configuration	Interconnect	CPU	CC	GPU	Arch	GPU SP/DP Flops	GPU Mem	GPU MB/W	Rtm.
P100-DGX-1	Single node, 8 GPUs	NVLink-V1	Intel Xeon E5-2698	gcc-4.8.4	Tesla-P100	Pascal	10609/5304	16GB HBM2	732 GB/s	8.0
V100-DGX-1	Single node, 8 GPUs	NVLink-V2	Intel Xeon E5-2698	gcc-5.4.0	Tesla-V100	Volta	14899/7450	16GB HBM2	900 GB/s	9.0
SummitDev	54 nodes, 4 GPUs/node	IB-EDR&NVLink-V1	IBM Power-8 NVL	xlc-13.1.6	Tesla-P100	Pascal	10609/5304	16GB HBM2	732 GB/s	8.0

TABLE II: Tartan Microbenchmarks.

Scaling	Communication	Interconnect	Metrics	Description
Scale-up	Peer-to-Peer	PCI-e, NVLink-V1 and V2	Latency, bandwidth and efficiency	Developed based on *p2pBandwidthLatencyTest* from CUDA SDK [12]
Scale-up	Collective	PCI-e, NVLink-V1 and V2	Latency, bandwidth and efficiency	Developed based on *nccl-tests* [13] linking to NCCL-V1 [14] and V2 [15]
Scale-out	Peer-to-Peer	InfiniBand with GPUDirect-RDMA	Latency, bandwidth and efficiency	Developed based on *MPI-GPU-BW* [16]
Scale-out	Collective	InfiniBand with GPUDirect-RDMA	Latency, bandwidth and efficiency	Developed based on *nccl-tests* [13] linking to NCCL-V2 [15]

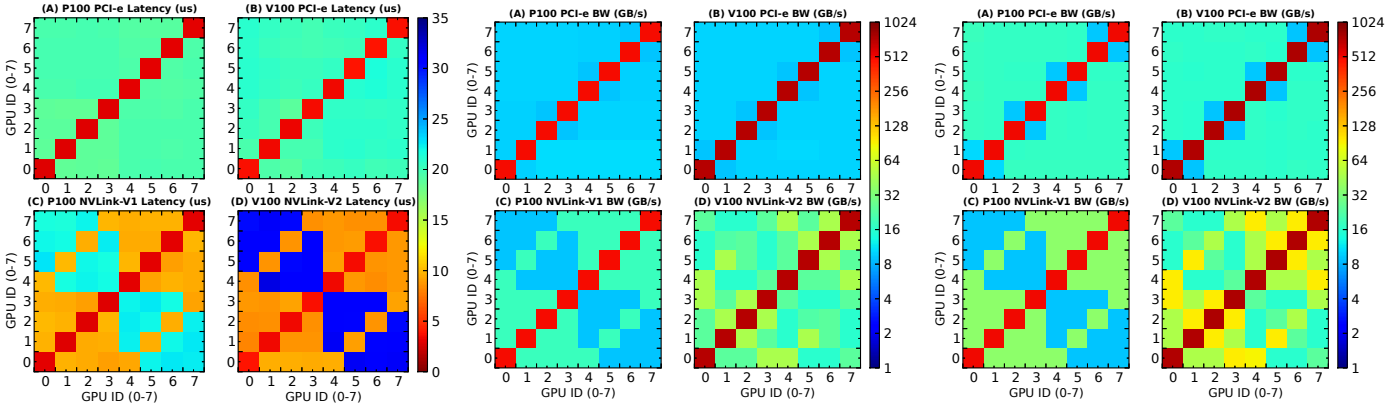

Fig. 2: Intra-node P2P communication latency. The red blocks along the anti-diagonal are local communication. The fact that other blocks are all green in (A) and (B) indicate that no NUMA effect appears on PCI-e for latency. The orange and blue blocks in (C) and (D) are refer to neighbor nodes and remote nodes respectively which exhibit clear disparity.

Fig. 3: Intra-node P2P unidirection bandwidth. Although not very obvious, we can see 2x2 blocks in (A) and (B) along anti-diagonal, which indicates the anti-locality NUMA effect for unidirection bandwidth on PCI-e. (C) and (D) confirm NUMA among neighbors and remote nodes. The other two types of NUMA for NVLink-V2 are not quite clear in (D). They are more obvious for bidirection bandwidth.

Fig. 4: Intra-node P2P bidirectional bandwidth. The 2x2 blue/red blocks along anti-diagonal in (A) and (B) clearly illustrate the anti-locality NUMA effect on PCI-e. In (D), the yellow blocks compared with the green blocks show the NUMA among neighboring nodes. Meanwhile, the light green blocks along the anti-diagonal (not quite obvious though) imply the existence of NUMA among remote nodes.

III. Interconnect Evaluation via Tartan Microbenchmarking

We evaluate the basic characteristics of the four GPU interconnects using the platforms listed in Table I, focusing on both **Peer-to-Peer (P2P)** and **Collective (CL)** communication patterns. For intra-node P2P, we especially concentrate on assessing the new node-level NVLink technology in terms of topology, latency, bandwidth, efficiency on message size and NUMA effects. For inter-node P2P, we discuss properties such as latency, bandwidth and efficiency on message size. Finally, we leverage the NCCL library to evaluate the collective communication patterns on both intra and inter-node scenarios. The microbenchmarks used for measuring latency, bandwidth and efficiency on message size are listed in Table II. We use *cudaEvent* to measure the latency and calculate the bandwidth.

A. Intra-Node P2P Communication

1) Latency: Figure 2 illustrates the *start-up communication latency* (known as the *raw latency* for transmitting the shortest message) among arbitrary pair of GPUs via PCI-e and NVLink for the P100 and V100 DGX-1 platforms (Table I), respectively. As mentioned in Section II-A, NVLink is not P2P self-routed. For GPUs that are not directly connected by NVLink (e.g., G0 and G5 in Figure 1), there are two routing paths that only require a single transit (e.g., from G0 to G5, either G1 or G4 can be specified as the router). In such scenarios, Figure 2 essentially shows the path exhibiting shorter latency.

PCI-e Latency: Figure 2-(A) and (B) demonstrate that the communication latency for accessing different pairs of GPUs via PCI-e are similar, implying that no NUMA effects appear in latency through PCI-e. In other words, the three types of latency by going through one PCI-e switch (e.g., G0 and G1), across local CPU socket (e.g., G0 and G2), and across the QPI bridge (e.g., G0 and G6) in Figure 1 are roughly the same. Meanwhile, comparing Figure 2-(A) and (B), the PCI-e latency is increased slightly (e.g., from green to light blue) from P100-DGX-1 to V100-DGX-1. As the bandwidth keeps unchanged, this may suggest a deeper communication pipeline design in V100-DGX-1 with Little's Law [17].

NVLink Latency: Compared to PCI-e (Figure 2-(A) and (B)), NVLink (Figure 2-(C) and (D)) demonstrates significant NUMA effects. For nodes that are directly connected, the latency is around $9\mu s$; for nodes that require manual routing, the latency is increased by about $2\times$ for P100-DGX-1 and $3\times$ for V100-DGX-1. We also observe that NVLink-V2 exhibits obvious higher latency than NVLink-V1, potentially due to a deeper communication pipeline or lower operating frequency.

2) Bandwidth: Figure 3 and 4 illustrate the unidirection and bidirection sustainable bandwidth for PCI-e and NVLink of the two DGX-1 platforms, respectively.

PCI-e Unidirection Bandwidth: In Figure 3-(A) and (B), we can observe slight NUMA effects on PCI-e accesses: GPUs sharing the same PCI-e switch (e.g., G2 and G3 in Figure 1) exhibit lower bandwidth. For other GPUs, no matter whether sharing the same socket, the bandwidth appears to be the same.

NVLink Unidirection Bandwidth: The NVLink scenario is more complicated. For NVLink-V1 in Figure 3-(C), there are three connection types: local access (i.e., the same GPU), neighboring nodes connected by NVLink, and remote nodes requiring additional routing, corresponding to the topology

illustrated in Figure 1-(A). For NVLink-V2 in Figure 3-(D), there are four connection types: local access, close neighboring nodes connected by dual NVLinks (i.e., the "backbone ring"), general neighboring nodes connected by one NVLink, and remote nodes, corresponding to the topology in Figure 1-(B). As such, there are three types of NUMA effects for NVLink:

- NUMA among neighbors and remote nodes for NVLink-V1 and V2 on both latency and bandwidth.
- NUMA among neighbors for NVLink-V2. This is due to different number of links (either 1 or 2) to connect neighboring nodes in V100-DGX-1. Basically, the latency remain similar but these two types of links introduce bandwidth difference.
- NUMA among remote nodes for NVLink-V2. This is caused by the choice of routing. Figure 3-(C) and (D) show the bandwidth disparity for choosing different routing methods.

PCI-e Bidirection Bandwidth: The PCI-e NUMA effects on bidirectional bandwidth for GPUs sharing the same PCI-e switch (i.e., Figure 4-(A) and (B)) are much more prominent than on that for unidirectional bandwidth. The PCI-e NUMA effect here is an interesting novel observation: it describes a scenario that *nearby access presenting lower performance than remote access.* We call such a NUMA effect as **anti-locality**. To the best of our knowledge, few existing work have ever discussed such a phenomenon, without mentioning leveraging it for performance tuning. The anti-locality effect is possibly due to the unbalanced physical signal paths on some of the PCIe-switch chipsets [18]. Note, such a PCI-e anti-locality effect is only observed for bandwidth rather than latency.

NVLink Bidirection Bandwidth: The three NVLink NUMA effects for bidirectional bandwidth are much more obvious than that for unidirectional bandwidth, as described in the caption of Figure 4.

3) Routing: We further explore the NUMA effects on routing choices for remote GPU access through NVLink. For demonstration purposes, we take G0 in Figure 1 as the source node for P2P communication. There are three remote nodes for G0: G5, G6 and G7. From G0 to G5, either G1 or G4 can be specified for routing. From G0 to G6, either G2 or G4 can be selected; and from G0 to G7, either G3 or G4 can be selected. We use microbenchmarks from Tartan to measure the latency, unidirection bandwidth and bidirection bandwidth of each routing path from G0 to G5, G6, G7 respectively on both DGX-1 platforms. Figure 5 shows the results for unidirection and bidirection bandwidth. As can be seen, for NVLink-V1 in P100-DGX-1, there are no NUMA effects; all the bars appear in the same height. This is because the NVLinks in P100-DGX-1 are isomorphic — any connection incorporates just one link. However, for NVLink-V2 in V100-DGX-1, different scenarios emerge based on how many dual-bandwidth links a route goes through, e.g., the lowest bandwidth occurs from G0→G2→G6 while the highest is triggered by routing G0→G4→G7. Nevertheless, the latency remain similar for all scenarios (not shown in the figures).

4) Efficiency on Message Size: Figure 6 illustrates the intra-node P2P latency, unidirection and bidirection bandwidth with respect to message size via both PCI-e and NVLink.

Latency: The latency remains unchanged for data communication less than or equal to 16KB for P100-DGX-1 (except local access). For V100-DGX-1, this value increases to 64KB, suggesting higher link bandwidth to saturate and deeper communication pipeline on NVLink-V2.

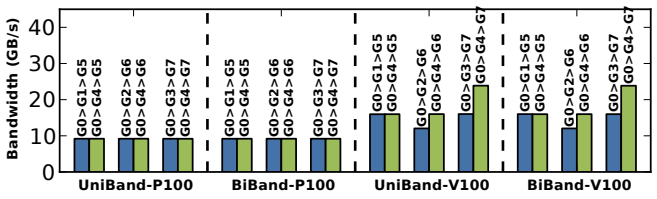

Fig. 5: NUMA effect on routing choices for remote GPU access via NVLink.

Bandwidth: For unidirection bandwidth, it can be seen that the interconnect starts to saturate at about 2^{22} =4MB for both NVLink-V1 and V2, implying that to reach the optimal sustainable bandwidth of the interconnect, one needs at least 4MB data to transmit at a time. This is also true for bidirection bandwidth in Figure 6-(E) and (F). In addition, observing the fact that the latency starts to increase at 16KB and 64KB, but the bandwidth begins to saturate at 4MB imply that from 64KB, we suffer from extra delay, but still gain overall bandwidth improvement until the bandwidth saturation point. Furthermore, local access bandwidth stagger at about 4MB for both NVLink-V1 and V2 due to exceeding page boundary.

B. Intra-Node Collective Communication

Different from P2P communication which contains one sender and one receiver, collective communication (CL) involves multiple senders and receivers so it is much more complicated. CL generally follow certain patterns, including broadcast, scatter, gather, all-gather, reduce, all-reduce, all-to-all, etc. It is extensively used in many key applications such as deep learning, molecular dynamics, graph analytics, etc.

Efficiently implementing CL communication is challenging because (a) it needs to understand the underlying hardware network topology in order to enable orchestrated mapping; (b) it needs to handle the issue of synchronization, overlapping and deadlock; and (c) performance metrics can differ according to application features (e.g., latency-oriented for small transfers but bandwidth-oriented for large transfers). To relieve some of these burdens from the users, NVIDIA provides a Collective Communication Library (NCCL) [14], [15] using similar primitives as MPI collectives (Correspondingly, AMD offers RCCL [19]). Currently, NCCL supports five CL patterns: *broadcast, all-gather, reduce, all-reduce*, and *reduce-scatter*.

To offer the maximum bandwidth, NCCL constructs ring network among the communication participants so that broadcasting and reduction can be efficiently realized by partitioning data into small chunks, and transmitting them along the ring in a pipeline fashion. It is claimed that the ring algorithm can provide near optimal bandwidth for most of the standard CL operations and can be easily applied to various network topology [3]. Figure 7 describes how the ring-network is constructed for PCI-e, NVLink-V1 and NVLink-V2, respectively.

There are two versions of the NCCL library: NCCL-V1 [14] is opensource but only supports intra-node PCI-e/QPI interconnect network. NCCL-V2 [14] is closed-source but supports NVLink, PCI-e/QPI, IB and IP networks and can automatically integrate them to maximize overall bandwidth. Although the combination improves overall performance, it also introduces difficulty when comparing PCI-e and NVLink on CL communication. Consequently, to evaluate PCI-e and NVLink separately, NCCL-V1 is employed for PCI-e CL evaluation while NCCL-V2 is adopted for NVLink CL evaluation.

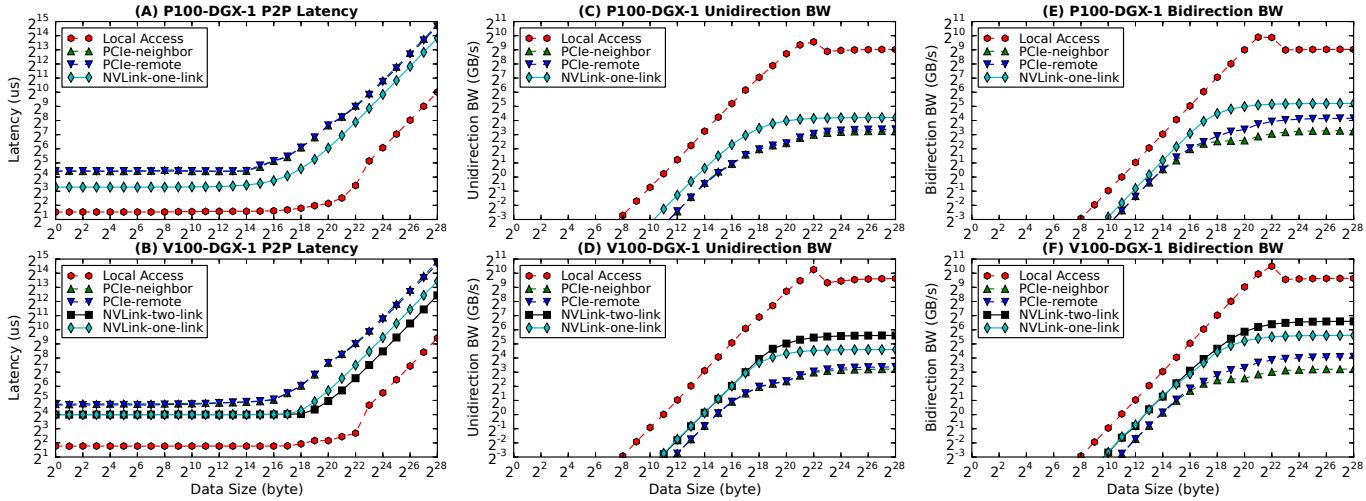

Fig. 6: Intra-node P2P communication latency, unidirection and bidirection bandwidth with increased message size via PCI-e and NVLink for DGX-1.

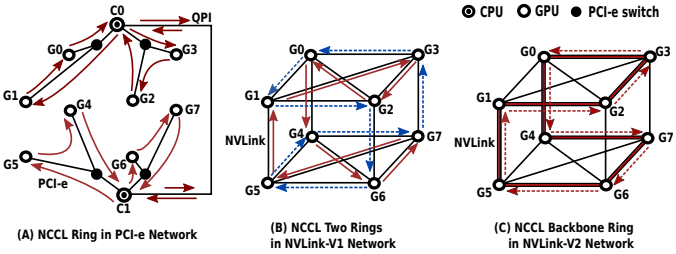

Fig. 7: NCCL Rings for PCI-e, NVLink-V1 and NVLink-V2 interconnect. (A) for PCI-e, the ring is to traverse the binary-tree network; (B) for NVLink-V1, there are two independent rings, marked in red-solid line and blue-dash line; and (C) for NVLink-V2, the lines with 2 links form a fast ring (i.e., the backbone network) while the lines with 1 link form a slow ring.

Figure 8 illustrates CL communication startup latency with respect to the number of GPUs involved for NCCL-V1 and V2 on the two DGX-1 machines respectively, corresponding to PCI-e/QPI and NVLink. We have the following observations: (1) latency increases with participating GPUs almost in a linear fashion; (2) comparing (A) and (C), (B) and (D), the behaviors of NCCL-V1 and V2 on the two DGX platforms are similar; (3) comparing (A) (B) with (C) (D), the latency of NVLink increases faster than PCI-e (except all-reduce), while NVLink-V2 increases faster than NVLink-V1; (4) for PCI-e, *all_reduce* shows the largest latency. The disalignment of the curves in (B) and (D) for odd number of GPUs (e.g., 3, 5) is likely due to the P2P NUMA effects in NVLink-V2 (Section III-A).

Figure 9 shows CL's sustainable communication bandwidth with increased number of GPUs under 1GB payload. As can be seen, (1) for PCI-e, CL bandwidth decreases with more GPUs, which is due to bus contention in a tree-network (Figure 7-(A)); (2) for NVLink, however, CL bandwidth essentially increases with more GPUs due to more connected links in a hypercube mesh network, see Figure 7-(B) and (C); (3) PCI-e and NVLink behavior on P100-DGX-1 and V100-DGX-1 are in similar trend. However, NVLink-V2 exhibits significantly better bandwidth with 4 GPUs ($\sim$1.6$\times$) and 8 GPUs ($\sim$2$\times$) compared to NVLink-V1, showing the strength of dual-links and backbone ring (Figure 7); (4) the NUMA effects appear quite significant with 5 GPUs, implying that there may be a congestion when forming a NCCL ring among 5 GPUs. One should avoid adopting 5 GPUs in their application setup.

Figure 10 shows CL bandwidth with respect to message size increasing from 8B to 1GB for 8 GPUs. As can be seen, for PCI-e, CL-bandwidth saturates at about $2^{24} = 16$MB; whereas for NVLink, bandwidth saturates around $2^{28} = 256$MB. Again, the five CL patterns exhibit similar trend in terms of bandwidth when scaling the message size.

C. Inter-Node P2P Communication via GPUDirect

We measure the latency and bandwidth of inter-node P2P communication on *SummitDev Supercomputer* [20] from *Oak Ridge National Laboratory (ORNL)*. SummitDev is a three cabinet system with 54 nodes in total. Each node contains two IBM Power-8 CPUs and four NVIDIA P100 GPUs. Both Inter-GPU and CPU-GPU connections are NVLink-V1. SummitDev is built as an early prototype system for ORNL's pre-exascale *Summit Supercomputer* [21]. SummitDev supports GPUDirect.

For inter-node P2P, we conduct our measurement under five configurations: (i) *GPUDirect-RDMA* is to directly access GPU memory among nodes with GPUDirect enabled ("*GPUDirect*" here refers to a system option, see the supplementary file); (ii) *PinnedMem-GPUDirect* is to first copy data from GPU memory to the pinned host memory, then transfer the data to another node's pinned host memory and finally copy to the targeted GPU memory, with GPUDirect enabled; (iii) *PinnedMem* is similar to (ii) but with GPUDirect disabled; (iv) *UnpinnedMem-GPUDirect* is to copy via host unpinned memory with GPUDirect enabled; and (v) *UnpinnedMem* is analogous to (iv) but with GPUDirect disabled.

The measured latency and bandwidth with respect to message size (from 4B to 1GB) under the five testing scenarios are illustrated in Figure 11. For better illustration, the latency curve uses log-scale Y-axis while bandwidth uses normal-scale Y. From Figure 11, we draw the following observations: (i) Until $2^{12} = 4$KB, there is little difference among the five curves in terms of both latency and bandwidth; (ii) Different from conventional wisdom, GPUDirect-RDMA shows the worst performance in the range from 4KB to 64KB for latency and from 4KB to 256KB for bandwidth, especially at 32KB. This is possibly due to limitations in some chipsets for P2P access through the CPU/IOH [22]; (iii) From 4MB on, GPUDirect-RDMA shows its advantage on bandwidth and obtains its optimal bandwidth — 12GB/s at 64MB. However, this is still lower than the PinnedMem-GPUDirect scheme, which demonstrates more than 14GB/s sustainable bandwidth with large message sizes; (v) it is also interesting to observe that the bandwidth of GPUDirect-RDMA actually degrades dramatically after 64MB, implying that breaking large mes-

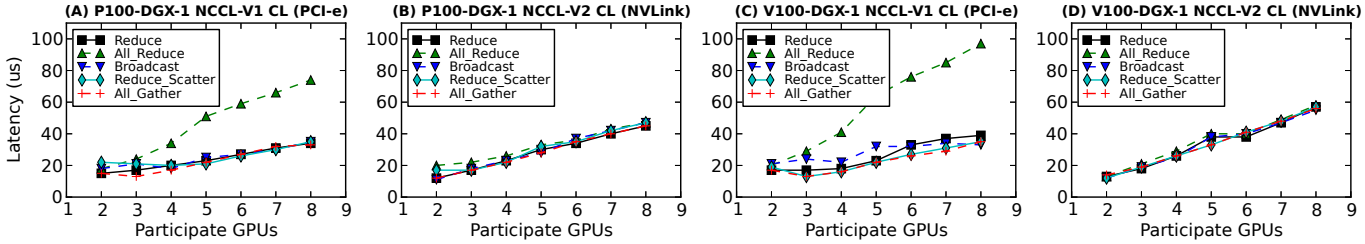

Fig. 8: Intra-node CL communication latency with variable participant GPUs for NCCL-V1 (PCI-e/QPI) and NCCL-V2 (NVLink-V1/2).

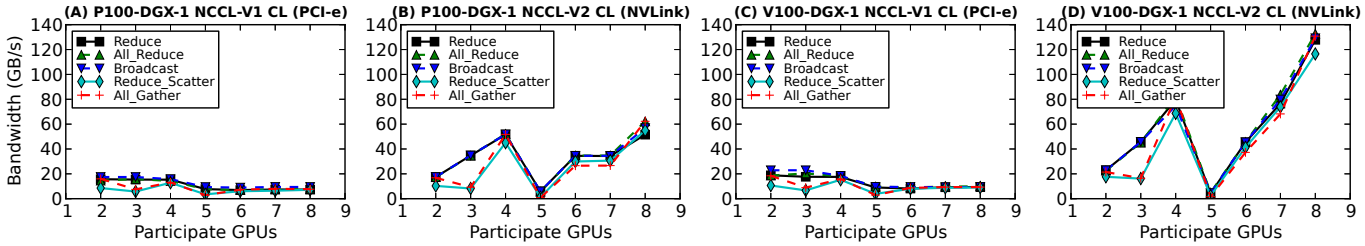

Fig. 9: Intra-node CL communication bandwidth with variable participant GPUs for NCCL-V1 (PCI-e/QPI) and NCCL-V2 (NVLink-V1/2).

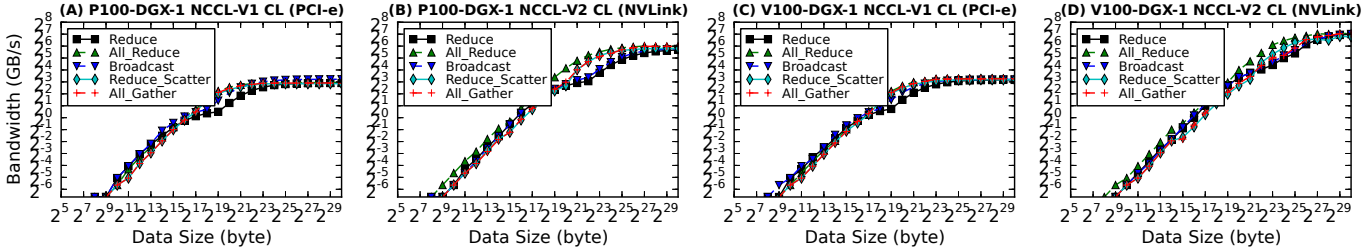

Fig. 10: Intra-node CL communication bandwidth for 8 GPUs with increased message size for NCCL-V1 (PCI-e/QPI) and NCCL-V2 (NVLink-V1/2).

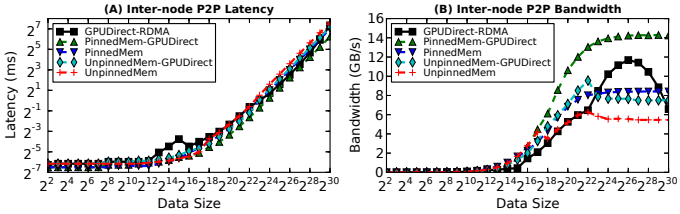

Fig. 11: Inter-node P2P latency and bandwidth with increased message size.

sage sizes into multiples of 64MB could be a better way to transfer in practice.

D. Inter-Node Collective Communication via GPUDirect

Regarding inter-node CL communication, we measure the latency and bandwidth with respect to the number of participant nodes on SummitDev. We tune the number of nodes from 2 to 8, with 1 GPU per node being utilized. Similarly, the startup latency is measured with 4B data transfer while the sustainable bandwidth is measured with sufficiently large data transfer (e.g., 1GB). The results are shown in Figure 12-(A) and (B). The bandwidth change with different message sizes is shown in Figure 12-(C). We tried to demonstrate the difference between enabling and disabling GPUDirect, but found that the results are in fact very similar. We suspect that GPUDirect-RDMA is internally enabled in NCCL-V2. As shown in Figure 12-(A), for the IB fat-tree network, the latency change for performing the five CL operations remains flat when scaling the number of nodes, except for *all-reduce*. In terms of bandwidth in Figure 12-(B), similar to NVLink (Figure 9-(B) and (D)), strong NUMA effects emerge under 3 and 5 nodes. However, the bandwidth overall remains unchanged. This is different from the bandwidth scenarios exhibited by both PCI-e (decreasing) and NVLink (increasing) in the inter-node P2P communication study. Finally, the bandwidth for the five CL operations converge and saturate around 32MB message size, demonstrating around 13GB/s sustainable peak bandwidth.

The microbenchmarking results exhibit some basic characteristics of modern GPU interconnects. However, in terms of real multi-GPU applications, their impact remains unknown. Since no multi-GPU benchmark suite is currently available yet, in the next section, we propose 14 applications in Tartan and use them to further evaluate GPU interconnects in Section V.

IV. Tartan Applications

In this section, we describe the applications in Tartan, which contain both scale-up and scale-out applications. To construct Tartan, we have proposed the following guidelines when collecting candidates: (i) the code can be compiled and executed properly in a multi-GPU environment; (ii) the code should be opensource and free to distribute; (iii) the code has little dependency on external library; (iv) the selected applications are representative and span across various domains; (v) the applications employ different communication patterns; and (vi) the applications are not hard-coded to only run on a fixed number of GPUs; (vii) the code should be relatively simple, and easy to tune. Under these constraints, we conducted code evaluation on over 50 candidates and narrowed down to 14 of them: seven scale-up applications and seven scale-out applications, as listed in Table III.

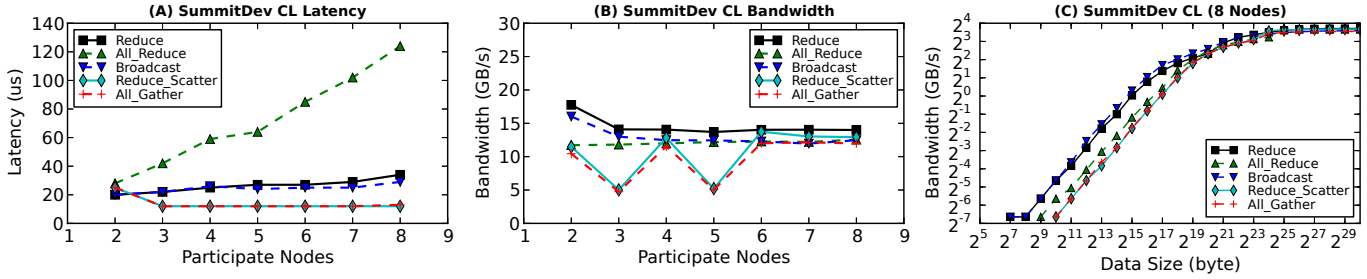

Fig. 12: Inter-node CL communication latency and bandwidth with variable participant nodes, as well as bandwidth for 8 nodes with increased message size.

To process the initial code, we first ported them into the unified Tartan compilation and execution environment with all the dependent libraries properly integrated. Then, we tracked the communication procedures in each application and conducted modification accordingly: for scale-up applications, we customize NVLink usage for each P2P and CL communication in a proper way (e.g., unified space, peer-to-peer copying or NCCL-based); for scale-out applications, we investigated each data-array participation in the inter-node communication, and generated the five communication implementations (Section III-B) for testing. Finally, we tuned the application interface so the problem size could be changed. We also prepare proper datasets and running scripts for strong and weak scaling on 1, 2, 4 and 8 GPUs (for intra-node scale-up applications) or GPU-integrated HPC nodes (for inter-node scale-out applications). Please refer to the supplementary file for details.

A. Intra-node Scale-up Applications

ConvNet2 [23] is a GPU implementation of convolution neural networks, which is capable of modeling various layer connectivity and network structures. ConvNet2 adopts single-node multi-GPU parallelization programming model and relies on GPU-unified-address to achieve P2P inter-GPU communication. Most of the computation and communication are issued asynchronously, thus can benefit from the overlapping of computation and communication. Based on our evaluation of the trace file printed by *nvprof*, $\sim$85% of the communication delay can be overlapped. However, this application is not working properly under 8 GPUs.

Cusimann [36], [24] implements parallel simulated annealing algorithm for global optimization. It uses OpenMP threads to manage each GPU accelerator in a single node. Master-slave parallel model is adopted here: CPU is the master, which handles the sequential parts while GPUs are the slaves, processing parallel portions. Communication occurs only between CPU and GPUs in the global reduction phase per iteration with relatively small data transmission volume. Note, all computation and communication are issued synchronously without overlapping. In the original design, at the end of each iteration, it is the CPU that gathers computation results from all GPUs, selects the optimal, and then broadcasts it to all GPUs for the next iteration. We observed that the CPU role here can be replaced by a GPU to benefit from a faster inter-GPU network, such as NVLink.

GMM [25] conducts multivariate data clustering via expectation maximization with Gaussian Mixture Model algorithm. It relies on OpenMP to manage GPUs. GMM also adopts the master-slave parallelization model similar to *Cusimann*, so there should be little inter-GPU communication. All GPU computation and communication are issued synchronously. By further investigating the code, we found that some of the data broadcasted by the CPU-master are essentially generated by GPU-0 in the previous iteration, while part of these arrays are not updated during the sequential phase. Thus, rather than broadcasting from CPU to each GPU, these arrays are directly copied from GPU-0 via NCCL broadcasting on NVLinks. In addition, the data merging phase (i.e., "*copy_cluster()*") on the CPU-side can essentially be moved to the GPU-side (e.g., GPU-0), and then broadcasted to other GPUs via NCCL.

Kmeans [26] performs Kmeans-Clustering for double-precision data on multi-GPUs attached to the same node. Similarly, the original version adopts master-slave communication pattern. Meanwhile, all computation and communication are issued synchronously. However, we observe that the sequential portions on the CPU side are only to aggregate results from GPU components, calculating the mean-value, and redistribute to each GPU. Thus, we apply NCCL *all-reduce* to enable data exchange among all GPUs and calculate their mean in each GPU locally. In this way, we convert CPU-GPU transfer to GPU-GPU transfer to benefit from NVLink and NCCL.

MonteCarlo [12] is a multi-GPU streaming application in which all the data are distributed by CPU to GPUs, and then returned to CPU after processing. Each result corresponds to a data input. There is no GPU-to-GPU communication so this application cannot directly benefit from NVLink. Computation and communication are issued asynchronously.

Planar [27] counts all permutations of the sequence 1,1,2,2,3,3,...,n,n in which the two occurrences of each m are separated by precisely m other numbers, and lines connecting all (m, m) pairs can be drawn without crossing. It runs a depth-first-search (DFS) and backtracking algorithm to check whether to open or close a pair at each position and how the pair can be connected. This application essentially follows a task model: each GPU fetches a task from the CPU stack, searches that branch and returns the search result. The communication only occurs between CPU and GPUs. All computation and communication are issued synchronously.

Trueke [28] performs exchange Monte Carlo for 3D random field Ising model. It originally uses OpenMP threads to manage GPUs. We observe that during two communication phases, when the data from GPU-0 is required to be distributed to the other GPUs, it first copies data to a CPU memory buffer, synchronizes, and then signals other GPUs to copy from that buffer. The computation and communication are issued synchronously. To leverage NVLink, we adopt NCCL broadcast to directly distribute data from GPU-0. Additionally, we reconstructed the function calls to amortize the initialization overhead of invoking NCCL.

B. Inter-node Scale-out Applications

B2rEqwp [29] is a 3D earthquake wave-propogation model simulation for GPUs based on 4-order finite difference method.

TABLE III: Tartan Benchmark Suite.

Application	Brief Description	abbr.	Domain	Comm	Scaling	Pattern	Source
ConvNet2	Convolution neural networks via data, model and hybrid parallelism	CNN	*Machine Learning*	CUDA	Scale-up	P2P	[23]
Cusimann	Global optimization via parallel simulated annealing algorithm	CSM	*Optimization*	OpenMP	Scale-up	CPU-GPU	[24]
GMM	Multivariate data clustering via Expectation Maximization with Gaussian mixture model	GMM	*Data Analysis*	OpenMP	Scale-up	CL-Broadcast	[25]
Kmeans	Kmeans clustering for double-precision data on multi-GPUs attached to the same node	KMN	*Data Analysis*	CUDA	Scale-up	CL-AllReduce	[26]
MonteCarlo	Monte Carlo option pricing from CUDA SDK	MTC	*Finance*	CUDA	Scale-up	CPU-GPU	[12]
Planar	Depth-first-search (DFS) and backtracking to solve Planar Langford's Sequences	PLN	*Number Theory*	CUDA	Scale-up	CPU-GPU	[27]
Trueke	Exchange Monte Carlo for 3D random field Ising model	TRK	*HPC Simulation*	OpenMP	Scale-up	CL-Broadcast	[28]
B2rEqwp	3D earthquake wave-propogation model simulation using 4-order finite difference method	BRQ	*HPC Simulation*	MPI	Scale-out	P2P	[29]
Diffusion	A multi-GPU implementation of 3D Heat Equation and inviscid Burgers' Equation	DFF	*HPC Simulation*	MPI	Scale-out	P2P	[30]
Lulesh	Livermore unstructured Lagrangian explicit shock hydrodynamics	LLH	*Molecular Dynamics*	MPI	Scale-out	P2P	[31]
CoMD	A reference implementation of classical molecular dynamics algorithms and workloads	CMD	*Molecular Dynamics*	MPI	Scale-out	P2P/CL	[32]
Prbench	Page rank computation by multi-GPUs	PRB	*Graph Processing*	MPI	Scale-out	P2P/CL	[33]
HIT	Simulating Homogeneous Isotropic Turbulence by solving Navier-Stokes equations in 3D	HIT	*HPC Simulation*	MPI	Scale-out	CL All-to-All	[34]
Matvec	Matrix multiplication via mpi-scatter, broadcast and gather	MAM	*Linear Algebra*	MPI	Scale-out	CL	[35]

The global 3D domain is partitioned into blocks, with each MPI node handles a block. In each iteration, *B2rEqwp* relies on MPI send/recv to exchange the halo region among blocks or nodes.

Diffusion [30] is an implementation simulating the Heat Equation and the inviscid Burgers' equation using CUDA-MPI. The package offers both single-GPU and multi-GPU versions for Burgers2d, Burgers3d, Diffusion2d and Diffusion3d. Here, we only use the multi-GPU version for Diffusion3d as a representative. Similar to *B2rEqwp*, *Diffusion* partitions the global domain into subdomains along the Z-axis, with each subdomain processed by an MPI node. It relies on MPI send/recv to exchange data in the boundary region via the left/right send/recv buffer.

Lulesh [31] stands for Livermore unstructured Lagrangian explicit shock hydrodynamics, which is a proxy application. It describes the motion of materials relative to each other when subject to forces. It partitions the global domain along the X-axis and exchange boundary data via MPI P2P communication.

CoMD [32] is also a proxy application which supports both cell-based and neighbor-lists methods for molecular dynamics. It is a reference implementation of the classical molecular dynamics algorithms and workloads [37]. It adopts a simple geometric decomposition to divide the total problem space into domains, each handled by an MPI rank. The MPI P2P communication is leveraged to exchange the halo regions.

Prbench [33] performs page-rank on multi-GPUs. Page-rank is an algorithm used by Google search to measure and rank the importance of website pages in the searching results. It implements the algorithm in a pipelined fashion. Data is exchanged by MPI send/recv P2P operations after each node computing *SpMV* locally.

HIT [34] is a parallel GPGPU code to simulate Homogeneous Isotropic Turbulence. It can solve the Navier-Stokes equations in a 3D periodic domain, following the algorithm described by Rogallo [38]. It partitions along the X-axis to MPI nodes and exchange data when transposing matrix before/after FFT via MPI All-to-All CL operations.

Matvec [35] is to perform a direct implementation of matrix multiplication on multi-node GPUs. The first matrix is partitioned along its rows among different MPI nodes and the second matrix is broadcasted to each node for multiplication. Three types of CL communication, including MPI_Scatter, MPI_Bcast and MPI_Gather are utilized in this application.

V. Evaluation

In this section, we use the Tartan application benchmark suite to evaluate the impact of GPU interconnect on multi-GPU execution performance. In particular, we focus on two aspects: (1) the impact of NVLink compared with PCI-e on scale-up applications; (2) the impact of GPUDirect-RDMA on scale-out applications. We perform two types of scaling measurement: (a) *strong scaling*, which fixes the problem size and measures the time reduction when increasing the number of GPUs; (b) *weak scaling*, which measures the time reduction when increasing the number of GPUs with fixed problem size per GPU. For overall performance, we use the entire application speedup (measured by CPU-side `time` command for whole application elapsed-time) as the performance metric, making a fair comparison across applications. For scale-up applications, we use the vendor-provided `nvprof` to measure the three types of GPU communication: *HostToDevice (H2D)*, *DeviceToHost (D2H)* and *DeviceToDevice (D2D)* in order to gain more knowledge about the underlying communication pattern. All the reported data points of the figures in this section are the average results of several times' execution.

A. Intra-node Scale-up

For intra-node scale-up scenarios, we evaluated the seven scale-up applications on P100-DGX-1 and V100-DGX-1, with and without NVLinks. Since many of these applications are hard-coded to leverage all the available GPUs in the system, we configure the system environment through *export CUDA_VISIBLE_DEVICES=x* to manipulate the number of GPUs being visible to the applications. Figure 13 and 14 illustrate the break out of the latency for the three types of communication (i.e., H2D, D2H, and D2D) regarding the original implementation (i.e., *Baseline*) and our modification (i.e., via *NCCL*) as described in Section IV-A, for intra-node strong and weak scaling on P100-DGX-1. The intention is that the original implementation will show the performance of PCI-e, while our modification would convert a big portion of CPU-GPU communication to GPU-GPU communication, so as to show the performance gain from NVLink. Figure 15 and 16 show the results on V100-DGX-1. The performance change of the entire application is given in the supplementary file due to space limitation.

Impact of NVLink. Although our observation from microbenchmarking in Section III show that NVLink can significantly improve inter-GPU communication efficiency, based on these figures, it is clear that those improvements do not directly transform into overall communication latency reduction, nor the whole application speedup (see the supplementary file); except `CSM` and `GMM`, there is not very significant difference between the Baseline and NCCL bars for both platforms. There are several reasons behind this. First, based on our experience on assessing the over 50 multi-GPU application candidates, most of those scale-up cases are based on master-slave programming model, where the CPU is the master, handling the sequential portions of code and GPUs are the slaves, processing the parallel portions. Under this model,

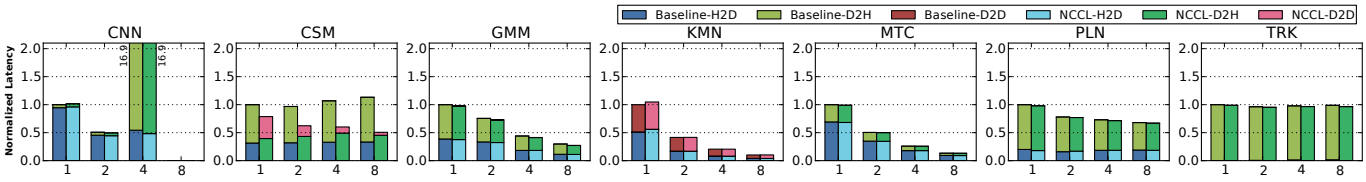

Fig. 13: Normalized latency reduction by NVLink-V1 and NCCL-V2 of strong scaling for single-node scaling-up on NVIDIA P100-DGX-1.

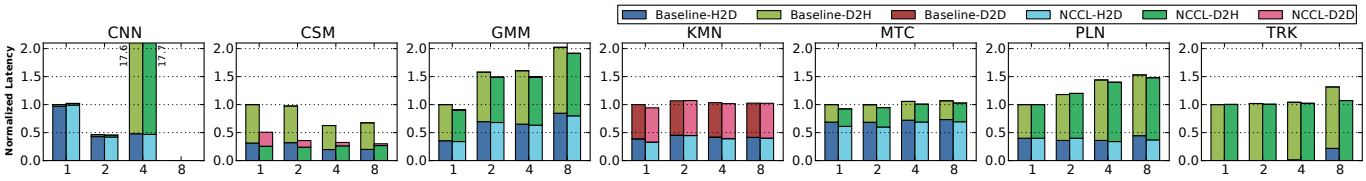

Fig. 14: Normalized latency reduction by NVLink-V1 and NCCL-V2 of weak scaling for single-node scaling-up on NVIDIA P100-DGX-1.

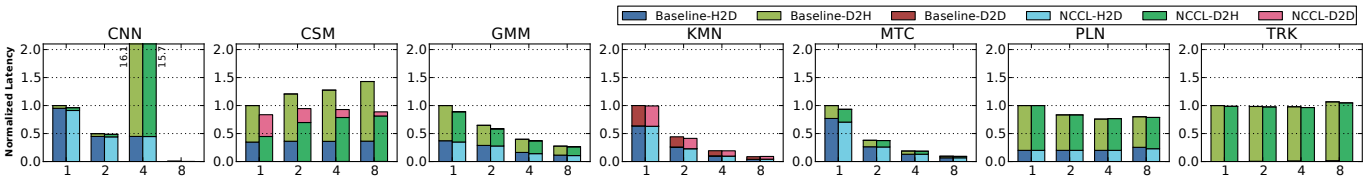

Fig. 15: Normalized latency reduction by NVLink-V2 and NCCL-V2 of strong scaling for single-node scaling-up on NVIDIA V100-DGX-1.

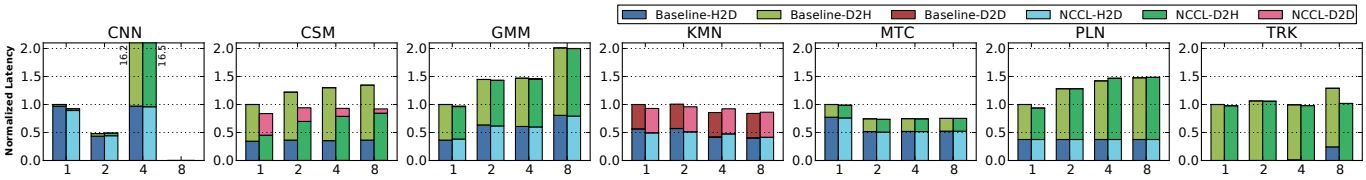

Fig. 16: Normalized latency reduction by NVLink-V2 and NCCL-V2 of weak scaling for single-node scaling-up on NVIDIA V100-DGX-1.

communication only occurs between CPU and GPUs; no inter-GPU transaction is presumed. In addition, the CPU-GPU communication is also highly optimized. This is true for `CSM`, `GMM`, `KMN`, `MTC`, `PLN` and `TRK`. For `CSM`, `PLN`, `GMM`, and `TRK`, we manage to convert some CPU-GPU communication into GPU-GPU communication through NCCL. As can be seen, for `CSM`, the effect is obvious: a large portion of the D2H and H2D communication is replaced by D2D. For `GMM`, although we gained $\sim$6% latency reduction, from the *nvprof* trace file, we did not observe any D2D communication transactions (but rather D2H and H2D) before/after NCCL's *BroadcastKernelSmall()* kernels. This is potentially caused by the flexible design strategy adopted by NCCL-V2 to efficiently leverage all available interconnect bandwidth (Section III-B). When the data size per transmission is small, it may not choose D2D communication via NVLink. Similar conditions also observed for `PLN` and `TRK`. For `KMN` and `MTC`, there is no GPU-GPU communication at all. The D2D in `KMN` is actually local data movement within the same GPU. `CNN` is another case. For `CNN`, the figures do not show bars under 8 GPUs because `CNN` implementation requires arbitrary data copying among arbitrary peers at runtime, which currently fails when 8 GPUs are utilized, since not every two GPUs are directly connected by NVLink under the current NVLink topology. As it internally uses *cudaMemcpyDefault* and *unified-space* for data copying across GPUs, NVLink is already internally enabled by CUDA for a NVLink-equipped machine such as DGX-1. We tried to modify the `CNN` code so that PCI-e can be essentially enforced but did not run correctly with success. This is why the two bars of `CNN` exhibit the same value for all four figures. It is also interesting to see that when more than 4 GPUs participant in the computation, the D2H communication increases dramatically, potentially due to the gradient merging overhead in the backpropagation. Secondly, since today's scale-up applications are mostly based on the master-slave programming model, communication often only accounts for a small fraction of the total execution time, let alone the inter-GPU communication which tended to be avoided previously when creating applications, thus hardly become the system bottleneck. Finally, employing NVLink (either P2P or CL) does introduce additional overhead (e.g., enable/disable peer access, routing, NCCL initialization, etc).

The performance scalability study can be found in the supplementary file. To summarize, NVLink has been reported to be helpful for accelerating modern deep-learning frameworks [39], [40]. However, regarding general GPGPU applications, without (i) replacing the underlying CPU-centric master-slave programming model by a more distributed parallelization model, or (ii) migrating the communication master role to a GPU (e.g., off-loading GPU communication control from CPU to GPU via techniques such as *NVSHMEM* [41]), optimized inter-GPU communication via faster intra-node GPU interconnect such as NVLinks can hardly become significant enough to lift the entire application's speedup. Therefore, we believe that this observation paves the road for developing interconnect-friendly programming models for multi-GPU scale-up scenarios so that faster interconnect (e.g., NVLinks) can truly play a role in improving the overall application efficiency.

B. Inter-node Scale-out

For inter-node scale-out scenarios, we run the seven scale-out applications on *SummitDev* (see Table I), with each MPI rank binding to a node using only a single GPU. Similar to the discussion in Section III-C, we measured the overall application performance under five scenarios: *GPUDirect-RDMA*, *PinnedMem-GPUDirect*, *PinnedMem*, *UnpinnedMem-GPUDirect* and *UnpinnedMem*.

Figure 17 and 18 illustrate the speedups with respect to single-node *UnpinnedMem* for strong and weak scaling tests, respectively. As can be seen, compared with the intra-node scale-up cases, the MPI-based inter-node scale-out applications exhibit much better scaling behavior in both strong and weak scaling tests, implying that compared with the intra-node fast interconnect, the inter-node network is much easier to become the system bottleneck. Improving inter-node network speed can lead to significant performance gain for multi-GPU applications. Regarding to GPUDirect-supported IB interconnect, we have the following observations: (i) Enabling GPUDirect can bring immediate performance enhancement, whether or not the transmitted data reside in CPU memory or GPU memory; (ii) Using pinned memory is also beneficial, especially in coordination with GPUDirect enabled; (iii) GPUDirect+RDMA can be especially helpful in certain applications (e.g., `BRQ` and `MAM`) where the benefits from avoiding allocating and buffering in system memory exceeds the overhead for handling RDMA at runtime (see the microbenchmarking in Section II-B).

Overall, for scale-out applications to benefit from a faster inter-node interconnect (e.g., IB-RDMA), the major difficulty is not from the hardware or the application, but from the communication abstract interfaces such as MPI. If a new MPI implementation can internally integrate the NCCL library, further harvesting multi-GPU interconnect performance (e.g., NVLink and IB-RDMA) can be much more easier. Again, initiating communication completely on the GPU side without CPU intervention (e.g., via *GPUDirect-Async* [42] or *GPU-triggered networking* [43]) may also be critical for good GPU performance delivery.

VI. Discussion

We discuss current design limitations and some potential research topics on GPU interconnect, particularly for NVLink. The current design of the fixed NVLink topology for V100-DGX-1 (Figure 1) aims at maximizing the overall performance under a variable number of GPUs (e.g., 2, 4, 8 GPUs). However, when a subset of particular number of GPUs are leveraged, other technology may offer additional benefit. For instance, the design in Figure 19-(A) can draw better connectivity when the whole machine is partitioned into two running subset, each with 4 GPUs (e.g., G0-G3 and G4-G7) while the design in Figure 19-(B) exhibits better connectivity when DGX-1 in partitioned into four running subsets (e.g., G0-G3, G1-G2, G4-G7, G5-G6). Ideally, if the NVLink connection can be reconfigured via some software approaches, the topology network could be adjusted dynamically according to an application's GPU demand as well as its potential NUMA preference, similar to the NUMA node configuration in Intel Xeon-Phi processors [44], [45].

NVLink is claimed to be transparent but in reality it is not that simple. Especially when two GPUs are not directly connected via NVLink, one has to explicitly specify the routing node and handle the synchronization based on the understanding of topology and mapping. For a multi-GPU application, how to smartly pick the optimal routing strategy at runtime can be rather complex (e.g., a routing node may focus on its own computation task when being requested for routing) and forms an interesting scheduling problem. A lower-level runtime abstraction could be a good solution to gain performance and hide such complexity from users. Additionally, when taking the interconnect technology into consideration, several interesting research problems emerge, e.g., resources sharing, task migration, memory consistency, and NUMA effects. Our hope is that this paper can serve as an initial attempt to motivate these potential future research.

As a future work, we are planning to: (i) port the Tartan benchmark suite to other vendor's GPUs [19], [11] via tools such as HIP [46]; (ii) build analytic models [47], [48], [45] for the multi-GPU interconnect, especially on bandwidth and NUMA effects, to facilitate scheduling-oriented performance tuning; (iii) continuously optimize the Tartan application code based on our experience on GPU performance optimization [49], [50], [51], [52], [53], [54], [55].

VII. Related Work

Intra-node GPU Computing. Spafford et al. [56] analyzed the NUMA effects in a multi-GPU node and provided optimization guidance. Kim et al. [57] proposed to rely on hybrid-memory-cubes (HMCs) to build a memory network for simplifying multi-GPU memory management and improving programmability. Wang et al. [58] presented a design to realize GPU-Aware MPI to support data communication among intra-node GPUs with standard MPI. Ben-Nun et al. [59] described an automatic multi-GPU partition framework to distribute workload based on their memory access patterns. Cabezas et al. [60] showed a software solution, including programming interfaces, compiler support and runtime, to partition GPU kernels for multi-GPU execution in a single node. Finally, Sun et al. [61] evaluated the potential performance benefit and tradeoffs of AMD's *Radeon Open Compute* (ROC) platform for *Heterogeneous System Architecture* (HSA).

Multi-node GPU Computing. For MPI-based multi-node GPU computing, Wang et al. [62] introduced a MPI design that integrates CUDA data movement transparently with MPI. Gysi et al. [63] proposed a hardware approach to overlap computation and communication in a GPU cluster. Klenk et al. [64] analyzed the exascale proxy applications on their communication patterns and proposed a matching algorithm for GPUs to comply with MPI constraints. Awan et al. [65] proposed a pipelined chain design for MPI broadcast collective operations on multi-GPU nodes to facilitate various deep learning frameworks.

Multi-GPU Interconnect Evaluation. Recently, several works have been proposed related to this topic [39], [66], [40]. However, most of them focus on evaluating and comparing different deep-learning frameworks, in terms of performance and scalability. Our work focuses on the interconnect impact on general multi-GPU execution in both scale-up and scale-out scenarios, and provides a practical benchmark suite for conducting these evaluation.

GPGPU Benchmark Suite. As GPGPUs have become increasingly popular in HPC due to their massive parallelism, several GPU benchmark suites have been proposed, including

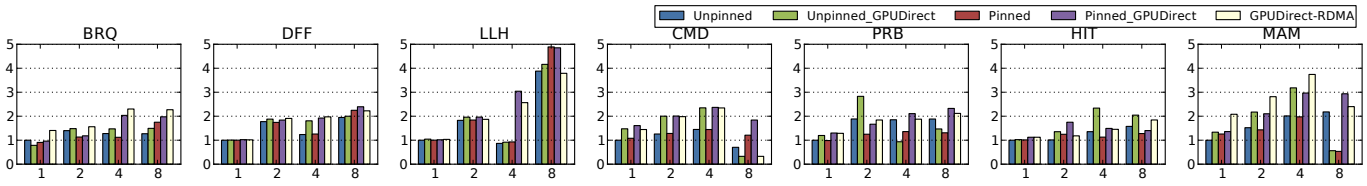

Fig. 17: Performance speedup by InfiniBand GPUDirect-RDMA of strong scaling for multi-node scaling-out on ORNL SummitDev.

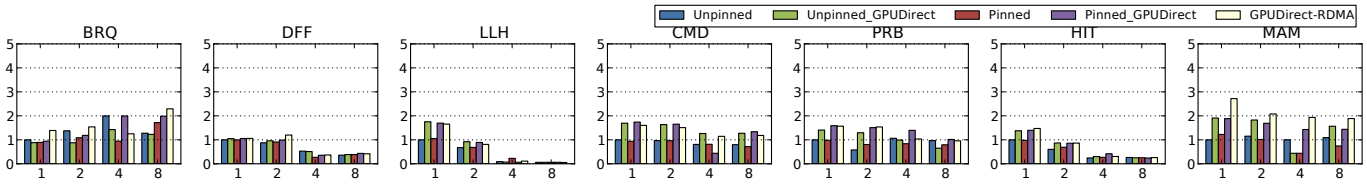

Fig. 18: Performance speedup by InfiniBand GPUDirect-RDMA of weak scaling for multi-node scaling-out on ORNL SummitDev.

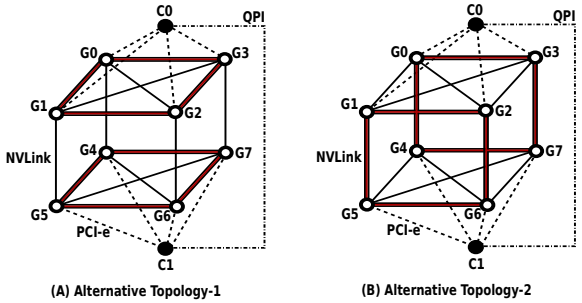

Fig. 19: Alternative V100-DGX-1 interconnect topology for NVLink-V2.

Rodinia [67], Parboil [68], SHOC [69], GPGPU-Sim [70], PolyBench [71], Lonestar [72], etc. However, none of these benchmarks were designed for multi-GPU purposes. Tartan fills this gap by providing microbenchmark and 14 applications for both intra-node and inter-node utilization. To the best of our knowledge, this is the first multi-GPU benchmark suite focusing on evaluating modern GPU interconnects.

VIII. Conclusion

In this paper, we proposed a multi-GPU benchmark suite named Tartan, which contains both microbenchmarks and practical scale-up and scale-out multi-GPU applications. We used Tartan to characterize and evaluate four types of modern GPU interconnects, including PCI-e, NVLink-V1, NVLink-V2 and InfiniBand with GPUDirect-RDMA. In particular, we investigated four types of NUMA effects and made several insightful observations for enabling practical optimization guidelines. Most importantly, the Tartan benchmark suite is opensource and aims to form a community efforts to support future development and maintenance. We hope Tartan and this evaluation study can help the HPC community to push forward multi-GPU research and development.

Acknowledgment

We thank all the anonymous reviewers for their constructive comments and suggestions for improving this work. This research was supported by the U.S. DOE Office of Science, Office of Advanced Scientific Computing Research, under award 66150: "CENATE - Center for Advanced Architecture Evaluation". This research was also supported by ECP Application Assessment program within the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05- 00OR22725. The Pacific Northwest National Laboratory (PNNL) is operated by Battelle for the U.S. Department of Energy under contract DE-AC05-76RL01830.

References

[1] 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.

[2] O. Fuhrer, T. Chadha, T. Hoefler, G. Kwasniewski, X. Lapillonne, D. Leutwyler, D. Lüthi, C. Osuna, C. Schär, T. C. Schulthess *et al.*, "Near-global climate simulation at 1 km resolution: establishing a performance baseline on 4888 GPUs with COSMO 5.0."

[3] NVIDIA, "NVIDIA DGX-1 System Architecture White Paper," 2015.

[4] S. Pabst, A. Koch, and W. Straßer, "Fast and scalable cpu/gpu collision detection for rigid and deformable surfaces," in *Computer Graphics Forum*, vol. 29, no. 5. Wiley Online Library, 2010, pp. 1605–1612.

[5] Q. Xu, H. Jeon, and M. Annavaram, "Graph processing on GPU: Where are the bottlenecks?" in *International Symposium on Workload Characterization (IISWC)*. IEEE, 2014.

[6] "The System Bottleneck Shifts to PCI-Express," https://www.nextplatform.com/2017/07/14/system-bottleneck-shifts-pci-express/.

[7] D. Foley and J. Danskin, "Ultra-Performance Pascal GPU and NVLink Interconnect," *IEEE Micro*, vol. 37, no. 2, pp. 7–17, 2017.

[8] P. Grun, "Introduction to infiniband for end users," *White paper, InfiniBand Trade Association*, 2010.

[9] NVIDIA, "CUDA Programming Guide," http://docs.nvidia.com/cuda/cuda-c-programming-guide, 2018.

[10] NVIDIA, "Developing a Linux Kernel Module using GPUDirect RDMA," http://docs.nvidia.com/cuda/gpudirect-rdma/index.html.

[11] AMD, "ROCm Driver RDMA Peer to Peer Support," https://github.com/RadeonOpenCompute/ROCnRDMA.

[12] NVIDIA, "CUDA SDK Code Samples," 2015.

[13] NVIDIA, "NCCL Tests," https://github.com/NVIDIA/nccl-tests.

[14] NVIDIA, "NVIDIA Collective Communications Library (NCCL-V1)," https://github.com/NVIDIA/nccl.

[15] NVIDIA, "NVIDIA Collective Communications Library (NCCL-V2)," https://developer.nvidia.com/nccl.

[16] "MPI-GPU-BW," https://github.com/frroberts/MPI-GPU-BW.

[17] J. D. Little, "A proof for the queuing formula: L= λ W," *Operations research*, 1961.

[18] Cirrascale, "Cirrascale SR3514: Unexpected Performance Inequality. Technical Brief M901A-092014."

[19] AMD, "ROCm Communication Collectives Library (RCCL)," https://github.com/ROCmSoftwarePlatform/rccl.

[20] OLCF, "Summit Early Access Development Platform," https://www.olcf.ornl.gov/for-users/system-user-guides/summitdev-quickstart-guide/.

[21] OLCF, "Summit: The Next Leap in Leadership-Class Computing Systems for Open Science," https://www.olcf.ornl.gov/for-users/system-user-guides/summit/.

[22] NVIDIA, "Developing a Linux Kernel Module using GPUDirect RDMA," https://docs.nvidia.com/cuda/gpudirect-rdma/index.html.

[23] Google, "High-Performance C++/CUDA Implementation of Convolutional Neural Networks Version-2," https://github.com/akrizhevsky/cuda-convnet2.

[24] A. M. F. Ferreiro, J. A. G. Rodrıguez, J. G. L. Salas, and C. V. Cendón, "CUSIMANN: An optimized simulated annealing software for GPUs," https://github.com/palmalcheg/cusimann.

[25] A. D. Pangborn, "Expectation Maximization with a Gaussian Mixture Model using CUDA," https://github.com/corv/cuda-gmm-multigpu.

[26] NVIDIA, "Kmeans Clustering with Multi-GPU Capabilities," https://github.com/NVIDIA/kmeans.

[27] B. Dimitrov, "Multi-GPU Code to Count all PLANAR Langford Sequences," https://github.com/boris-dimitrov/z4_planar_langford_multigpu.

[28] C. Navarro, "Multi-GPU Exchange Monte Carlo for 3D Random Field Ising Model ," https://github.com/crinavar/trueke.

[29] Z. Liu, "Efficient Large-scale Parallel Stencil Computation on Multi-Core and Multi-GPU Accelerated Clusters," https://github.com/lzhengchun/b2r.

[30] M. A. Diaz, "Multi-GPU (CUDA-MPI) baseline implementation of Heat Equation and the inviscid Burgers' equation," https://github.com/wme7/MultiGPU_AdvectionDiffusion.

[31] LLNL, "Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics," https://codesign.llnl.gov/lulesh.php.

[32] NVIDIA, "GPU implementation of classical molecular dynamics proxy application," https://github.com/NVIDIA/CoMD-CUDA.

[33] NVIDIA, "A CUDA implementation of the PageRank Pipeline Benchmark ," https://github.com/NVIDIA/PRBench.

[34] M. V. Martín, "HIT: a parallel GPGPU code to simulate Homogeneous Isotropic Turbulence," https://github.com/albertovelam/HIT_MPI.

[35] T. Agarwal, "Multi-GPU Matrix Multiplication using CUDA and MPI," https://github.com/tejaswiagarwal/multigpumatmul.

[36] A. M. F. Ferreiro, J. A. G. Rodrıguez, J. G. L. Salas, and C. V. Cendón, "CUSIMANN: An optimized simulated annealing software for GPUs."

[37] C. C. for Particle Applications (CoPA), "GPU implementation of classical molecular dynamics proxy application," https://github.com/ECP-copa/CoMD.

[38] R. Rogallo, "Numerical experiments in homogeneous turbulence, NASA Tech," *Memo*, vol. 81, p. 315, 1981.

[39] S. Shams, R. Platania, K. Lee, and S.-J. Park, "Evaluation of Deep Learning Frameworks Over Different HPC Architectures," in *37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017.

[40] N. R. Tallent, N. A. Gawande, C. Siegel, A. Vishnu, and A. Hoisie, "Evaluating on-node gpu interconnects for deep learning workloads," in *International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*. Springer, 2017.

[41] S. Potluri, A. Goswami, D. Rossetti, C. Newburn, M. G. Venkata, and N. Imam, "Gpu-centric communication on nvidia gpu clusters with infiniband: A case study with openshmem," in *24th International Conference on High Performance Computing (HiPC)*. IEEE, 2017.

[42] E. Agostini, D. Rossetti, and S. Potluri, "Gpudirect async: Exploring gpu synchronous communication techniques for infiniband clusters," *Journal of Parallel and Distributed Computing*, vol. 114, pp. 28–45, 2018.

[43] M. LeBeane, K. Hamidouche, B. Benton, M. Breternitz, S. K. Reinhardt, and L. K. John, "GPU triggered networking for intra-kernel communications," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. ACM, 2017.

[44] J. Jeffers and J. Reinders, *Intel Xeon Phi coprocessor high performance programming*. Newnes, 2013.

[45] A. Li, W. Liu, M. R. Kristensen, B. Vinter, H. Wang, K. Hou, A. Marquez, and S. L. Song, "Exploring and analyzing the real impact of modern on-package memory on HPC scientific kernels," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. ACM, 2017.

[46] AMD, "HIP: Convert CUDA to Portable C++ Code," https://github.com/ROCm-Developer-Tools/HIP.

[47] A. Li, Y. Tay, A. Kumar, and H. Corporaal, "Transit: A visual analytical model for multithreaded machines," in *International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*. ACM, 2015.

[48] A. Li, S. L. Song, E. Brugel, A. Kumar, D. Chavarria-Miranda, and H. Corporaal, "X: A comprehensive analytic model for parallel machines," in *International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2016.

[49] A. Li, G.-J. van den Braak, A. Kumar, and H. Corporaal, "Adaptive and transparent cache bypassing for GPUs," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. ACM, 2015.

[50] A. Li, G.-J. van den Braak, H. Corporaal, and A. Kumar, "Fine-grained synchronizations and dataflow programming on GPUs," in *International Conference on Supercomputing (ICS)*. ACM, 2015.

[51] A. Li, S. L. Song, M. Wijtvliet, A. Kumar, and H. Corporaal, "SFU-driven transparent approximation acceleration on GPUs," in *International Conference on Supercomputing (ICS)*. ACM, 2016.

[52] A. Li, S. L. Song, A. Kumar, E. Z. Zhang, D. Chavarría-Miranda, and H. Corporaal, "Critical points based register-concurrency autotuning for GPUs," in *Conference on Design, Automation & Test in Europe*, 2016.

[53] W. Liu, A. Li, J. Hogg, I. S. Duff, and B. Vinter, "A synchronization-free algorithm for parallel sparse triangular solves," in *European Conference on Parallel Processing (EuroPar)*. Springer, 2016.

[54] A. Li, S. L. Song, W. Liu, X. Liu, A. Kumar, and H. Corporaal, "Locality-aware CTA clustering for modern GPUs," *ACM SIGOPS Operating Systems Review*, 2017.

[55] A. Li, W. Liu, L. Wang, K. Barker, and S. L. Song, "Warp-consolidation: A novel execution model for gpus," in *International Conference on Supercomputing (ICS)*. ACM, 2018.

[56] K. Spafford, J. S. Meredith, and J. S. Vetter, "Quantifying NUMA and contention effects in multi-GPU systems," in *Fourth Workshop on General Purpose Processing on Graphics Processing Units (GPGPU)*. ACM, 2011.

[57] G. Kim, M. Lee, J. Jeong, and J. Kim, "Multi-GPU system design with memory networks," in *47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2014.

[58] H. Wang, S. Potluri, D. Bureddy, C. Rosales, and D. K. Panda, "GPU-aware MPI on RDMA-enabled clusters: Design, implementation and evaluation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 10, pp. 2595–2605, 2014.

[59] T. Ben-Nun, E. Levy, A. Barak, and E. Rubin, "Memory access patterns: the missing piece of the multi-GPU puzzle," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. ACM, 2015.

[60] J. Cabezas, L. Vilanova, I. Gelado, T. B. Jablin, N. Navarro, and W.-m. W. Hwu, "Automatic parallelization of kernels in shared-memory multi-GPU nodes," in *International Conference on Supercomputing (SC)*. ACM, 2015.

[61] Y. Sun, S. Mukherjee, T. Baruah, S. Dong, J. Gutierrez, P. Mohan, and D. Kaeli, "Evaluating performance tradeoffs on the radeon open compute platform," in *International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2018.

[62] H. Wang, S. Potluri, M. Luo, A. K. Singh, S. Sur, and D. K. Panda, "MVAPICH2-GPU: optimized GPU to GPU communication for InfiniBand clusters," *Computer Science-Research and Development*, vol. 26, no. 3-4, p. 257, 2011.

[63] T. Gysi, J. Bär, and T. Hoefler, "dCUDA: hardware supported overlap of computation and communication," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE, 2016.

[64] B. Klenk, H. Fröening, H. Eberle, and L. Dennison, "Relaxations for high-performance message passing on massively parallel SIMT processors," in *International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2017.

[65] 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?" *arXiv preprint arXiv:1707.09414*, 2017.

[66] S. Shi and X. Chu, "Performance Modeling and Evaluation of Distributed Deep Learning Frameworks on GPUs," *arXiv preprint arXiv:1711.05979*, 2017.

[67] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 2009.

[68] J. A. Stratton, C. Rodrigues, I.-J. Sung, N. Obeid, L.-W. Chang, N. Anssari, G. D. Liu, and W.-M. W. Hwu, "Parboil: A revised benchmark suite for scientific and commercial throughput computing," *Center for Reliable and High-Performance Computing*, 2012.

[69] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter, "The Scalable Heterogeneous Computing (SHOC) Benchmark Suite," in *3rd Workshop on General-Purpose Computation on Graphics Processing Units (GPGPU)*. ACM, 2010.

[70] A. Bakhoda, G. L. Yuan, W. W. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2009.

[71] S. Grauer-Gray, L. Xu, R. Searles, S. Ayalasomayajula, and J. Cavazos, "Auto-tuning a high-level language targeted to GPU codes," in *Innovative Parallel Computing (InPar)*. IEEE, 2012.

[72] M. Burtscher, R. Nasre, and K. Pingali, "A Quantitative Study of Irregular Programs on GPUs," in *International Symposium on Workload Characterization (IISWC)*. IEEE, 2012.