Bandwidth Characterization of DeepSpeed on Distributed Large Language Model Training

Bagus Hanindhito
The University of Texas at Austin
Austin, Texas, USA
hanindhito@bagus.my.id

Bhavesh Patel
Dell Technologies
Round Rock, Texas, USA
bhavesh_a_patel@dell.com

Lizy K. John
The University of Texas at Austin
Austin, Texas, USA
ljohn@ece.utexas.edu

***Abstract*—The exponential growth of the training dataset and the size of the large language model (LLM) significantly outpaces the incremental memory capacity increase in the graphics processing units (GPUs). Thousands of GPUs are needed to handle state-of-the-art models, which require building an expensive AI GPU cluster that is out of reach for most researchers. This not only makes the cost to train the model more costly but also signifies the environmental impact. To improve the efficiency and scalability of existing infrastructure to handle increasingly demanding training tasks, Microsoft released DeepSpeed, an open-source optimization library for PyTorch that can easily be integrated into existing training flow with minimal code changes.**

This paper presents a comprehensive third-party evaluation of DeepSpeed for training GPT-2-like LLM on mainstream GPU clusters that are more accessible to everyone. The evaluation includes memory usage analysis and bandwidth characterization in addition to the achieved model size and the attained compute throughput to help compare horizontal and vertical scaling.

First, we examine the DeepSpeed ZeRO in single- and dual-node training against the popular distributed training libraries: PyTorch Distributed Data-Parallel (DDP) with data parallelism and Megatron-LM with data and model parallelism. While DDP achieves higher throughput due to less communication, the model size is limited to a single GPU memory capacity. In single-node training, Megatron-LM can fit a 4x larger model than the DDP, while ZeRO can handle a model with 0.8x-1.2x size of the Megatron-LM. Both Megatron-LM and ZeRO are reasonably competitive in terms of throughput. However, in dual-node training, Megatron-LM sees a significant drop in throughput due to the excessive inter-node communication, achieving only 25%-30% of the throughput offered by ZeRO. Secondly, we evaluate ZeRO-Offload to consolidate multi-node training into single-node. With CPU offloading, ZeRO-Offload allows single node to fit the largest model that can be handled on dual nodes with Megatron-LM while maintaining 57.8% higher throughput. Thirdly, we demonstrate that by using NVME offload on ZeRO-Infinity, we can fit model six times larger than previously possible in single node. Finally, we highlight the importance of NVME aggregate bandwidth as it significantly affects achieved throughput.

***Index Terms*—Bandwidth Characterization, Distributed Training, Large Language Model, Microsoft DeepSpeed, Megatron-LM**

I. Introduction

Following the neural network scaling laws [1–3], the Large Language Models (LLMs), which have been gaining traction in recent years [4–10], experience exponential growth in model size, a factor of 1000x between 2018 and 2020, as shown in Fig. 1, However, Graphics Processing Units (GPUs), popular accelerators in deep learning [11–16], only see a 5x increase in memory capacity during the same period [17, 18]. This

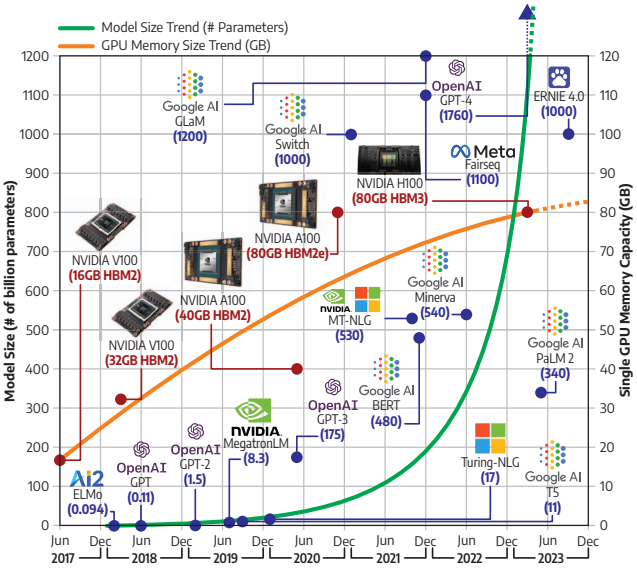

Fig. 1. The trend in the growth of Large Language Model size (in billion parameters) compared against the increase in single GPU memory capacity.

imbalance is expected to continue as Generative AI [19–22], demands even larger model [19, 23] to achieve target accuracy.

It is no longer possible to train state-of-the-art models using single GPU; distributed training across hundreds or thousands of GPUs are needed to get the aggregate compute, memory, and bandwidth. These require building expensive purpose-built AI GPU clusters, which are simply out of reach for many researchers. Training models becomes more expensive [24, 25] and gives significant impact to the environment [26–29].

Popular libraries for LLM distributed training includes PyTorch Distributed Data-Parallel (DDP) [30] and Megatron-LM [31–33]. DDP extracts data parallelism inherent in massive datasets to train LLM by replicating the model on each GPU, computing the gradients on subsets of datasets in each GPU, and synchronizing them across all GPUs. However, the model's size is limited to single GPU memory capacity. This is where Megatron-LM shines; not only does it extract data parallelism, but it also splits the model across multiple GPUs, achieving model parallelism for handling larger LLMs.

In this paper, we evaluate Microsoft DeepSpeed [34], a PyTorch-compatible optimization library, to train GPT-2-like LLM in comparison to DDP and Megatron-LM. DeepSpeed brings numerous optimizations with a minimal code change

2766-0486/24/$31.00 ©2024 IEEE
DOI 10.1109/ISPASS61541.2024.00031

to the existing training flow: ZeRO [35], ZeRO-Offload [36], and ZeRO-Infinity [37]. While publications from the developer of DeepSpeed already analyzed the model size and compute throughput [34–37], our work provides a more comprehensive third-party evaluation by adding bandwidth utilization characterization and memory usage analysis, which is extremely useful for comparing horizontal and vertical scaling. Unlike most previous studies that perform their evaluation in purpose-built AI GPU clusters, we aim for mainstream GPU clusters, which are more accessible to everyone, to perform the experiments. The objectives of our paper are the following:

- We build a small GPU cluster representing mainstream GPU clusters (Section III-A) and perform bandwidth stress-tests to identify communication bottlenecks (Section III-C).
- We evaluate DeepSpeed ZeRO on the achieved model size, compute throughput, memory usage, and bandwidth utilization in single- or dual-node training by comparing against DDP and Megatron-LM (Section IV).
- We investigate the effectiveness of DeepSpeed ZeRO-Offload in handling larger model sizes that were not previously possible by offloading parts of model states to CPU memory (Section V-A). We compare the compute throughput and bandwidth utilization to multi-node Megatron-LM.
- We explore the NVME offloading on DeepSpeed ZeRO-Infinity and its practicality to handle even larger model sizes (Section V-B and V-E). We identify the bottleneck and outline the efforts to improve the throughput.

The major insights of this paper are summarized as follows:

- We observed a significant degradation of attained bandwidth for data transfer between two I/O interfaces in AMD CPUs and hypothesized that it is due to the contention between two sets of I/O Serializer-Deserializer (SerDes).
- DDP achieves higher compute throughput in both single- and dual-node training due to lower communication.
- In single-node training, Megatron-LM can fit a model 4x larger than DDP while utilizing 300% more NVLink bandwidth. On the other hand, DeepSpeed ZeRO can fit a model with 0.8x-1.2x Megatron-LM size while utilizing a slightly higher bandwidth than DDP. Both Megatron-LM and ZeRO are fairly competitive in terms of compute throughput.
- In dual-node training, Megatron-LM can fit a model 8x larger than DDP with only 0.19x compute throughput due to the excessive inter-node communication. DeepSpeed ZeRO can fit a model with 0.56x-1.18x Megatron-LM size while having 3.26x-3.78x higher throughput.
- Offloading to CPU memory using DeepSpeed ZeRO-Offload allows single node to fit the largest model size Megatron-LM can handle in dual nodes while giving 1.58x higher throughput.
- Finally, using ZeRO-Infinity with NVME offload, we can fit a model six times larger than what the Megatron-LM can handle in a single node. This significantly reduces infrastructure costs and allows many researchers to have access to state-of-the-art models. Further, we highlight the importance of the aggregate bandwidth of NVME and the data placement on the achieved throughput.

II. Background

A. Hardware and ML Workload Imbalance Trend

Neural network scaling laws [1–3] relate the size of the training dataset, the model's size, the cost of training the model, and the model's performance (accuracy). Models that are exposed to larger high-quality datasets during training achieve higher accuracy [7, 38–42], driving dataset grows exponentially [8, 43]. In addition, larger model size also provides better accuracy [44, 45], fueling the trend in the exponential growth of model size [8, 46, 47], including the recently-popular large language model (LLM) [4–10, 48–51]. Nowadays, LLMs are used in many applications [52–72] by fine-tuning them for specific tasks [73]. The size of the LLMs grows exponentially: a factor of 1000 in two years from 94 million parameters ELMo (2018) [74] to 175 billion parameters GPT-3 (2020) [75]. The introduction of ChatGPT [76–78] marked the beginning of the Generative AI [19–22], which demands larger models [19, 23]. Recently released GPT-4 is estimated to have 1.76 trillion parameters [79].

On the other hand, Graphics Processing Units (GPUs), the popular accelerators for training neural networks [11–16, 80], only see a 5x increase in their memory capacity during the same period: 16 GB on NVIDIA Tesla V100 [17] released on June 2017 to 80 GB on NVIDIA A100 [18] released on November 2020. The successor, NVIDIA H100 [81], was released in March 2023 and still retains the same 80 GB memory size. State-of-the-art models no longer fit into a single GPU [82]; hundreds, even thousands of GPUs providing higher aggregate computational power, memory, and bandwidth [83] are required to handle these enormous models [84, 85].

These large numbers of GPUs require infrastructure (e.g., power, cooling, communication) built around them to provide the computation power needed to train the models. Purpose-built AI clusters (e.g., NVIDIA Selene [86]) are expensive, driving the cost to train the models higher [24, 25] and making them out of reach for many researchers. In addition, the energy required and the environmental impact become more concerning [26–29]. Furthermore, since most of the frameworks focus on utilizing the GPUs, other components (e.g., CPU, memory, NVME storage) are often left underutilized. However, they may be as expensive as GPUs, and thus, making the most out of them is compelling research [87–89].

B. Model Parallelism on Megatron-LM

Distributed training [90–94] is a crucial strategy for handling increasingly large datasets and model sizes by splitting the workloads into multiple processors (e.g., CPUs, GPUs). One of the popular methods is data parallelism (DP) [30, 95–98], which splits the dataset across multiple processors. Each processor holds the same copy of the model and performs the forward and backward passes [99] on a portion of the dataset independently. Then, each processor performs synchronization of gradients [100] or updated parameters [97]. However, exploiting DP alone limits the model size to the single GPU memory capacity; any larger will hurt performance due to the excessive data movement between CPU and GPU [101].

TABLE I
DEEPSPEED ZERO STAGE AND OFFLOAD CAPABILITY

Stage	Model State Partition			Offload Options			
	Optimizer	Gradient	Parameter	Optimizer		Parameter	
				CPU	NVME	CPU	NVME
0	DeepSpeed is disabled						
1	✔	-	-	✔	-	-	-
2	✔	✔	-	✔	-	-	-
3	✔	✔	✔	✔	✔	✔	✔

Model parallelism (MP) splits the model across multiple processors [102–105]. MP allows the aggregation of memory to handle larger model sizes and can be used together with DP. There are two implementations of MP: Pipeline Parallelism (PP) and Tensor Parallelism (TP). PP distributes each layer of the model to each processor [106–111], while TP slices each layer and distributes the chunks into multiple processors [112–115]. Although both PP and TP require major changes to the model implementation, some libraries provide built-in support for them with minimal code changes. Megatron-LM [31–33], a library developed by NVIDIA to train LLM, supports distributed training using TP and PP in addition to DP [116] by performing a few modifications to the existing PyTorch [117] transformer implementations.

C. Meet the Microsoft DeepSpeed

DeepSpeed is an open-source optimization library developed by Microsoft and is compatible with PyTorch [34]. It features novel memory optimization techniques called Zero Redundancy Optimizer (ZeRO) [35], which improves DP efficiency (called ZeRO-DP). ZeRO partitions the model states (optimizer states, gradients, model parameters) across data-parallel processes [118] with three stages as shown in Table I. ZeRO-1 and ZeRO-2 are promised to provide 4x and 8x memory reduction, respectively, with the same communication volume as the standard DP by partitioning optimizer states and gradients. ZeRO-3 partitions all model states and promises to have linear memory reduction depending on the degree of DP at the expense of 50% increase in communication volume.

Another notable feature is ZeRO-Offload, which allows offloading parts of the model states into CPU memory and performs optimizer computation in the CPU [36]. As shown in Table I, ZeRO-1 and ZeRO-2 support offloading optimizer states and gradients to CPU memory, while ZeRO-3 further adds support to offload model parameters to CPU memory. The ZeRO-3 offloading capability is extended by ZeRO-Infinity, allowing offloading model states to NVME storage in addition to CPU memory [37]. Further, DeepSpeed supports hybrid parallelism, including TP, PP, and DP [119].

DeepSpeed promises minimal code changes to the existing training flow in PyTorch. A JSON file containing the DeepSpeed configuration is supplied when running the training using the DeepSpeed launcher. DeepSpeed also provides Flops Profiler to give insight into the training performance [120].

III. METHODS

A. Hardware and Software Setup

1) Cluster setup: A small GPU cluster consisting of two identically configured Dell PowerEdge XE8545 [121] compute

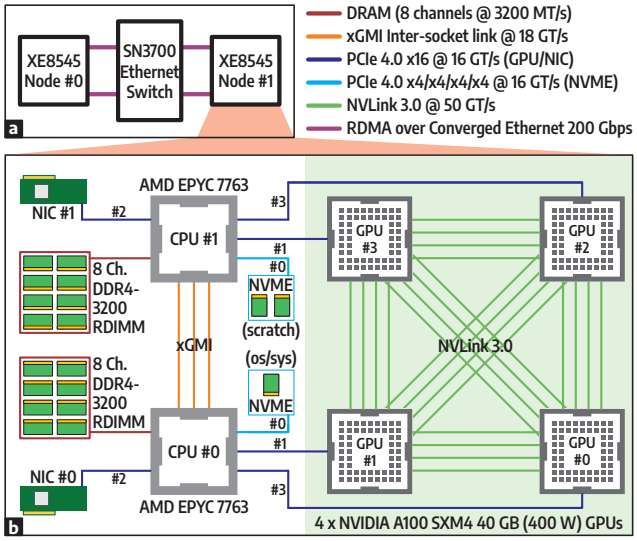

Fig. 2. The topology of small GPU cluster consisting of two XE8545 compute nodes and SN3700 switch (a) and the internal topology of each XE8545 (b).

TABLE II
HARDWARE AND SOFTWARE SETUP

Hardware Configuration	
Platform	Dell PowerEdge XE8545
CPU	2 × AMD EPYC 7763 CPUs (64 cores, 128 threads each)
Memory	16 × 64 GB DDR4-3200 ECC RDIMMs
GPU	4 × NVIDIA A100 SXM4 40 GB 400W GPUs
NVME	3 × Intel D7-P5600 3.2 TB PCIe 4.0 (1 OS, 2 scratch)
NIC	2 x NVIDIA ConnectX-6 NICs (200 Gbps)
Software Configuration	
Firmware	XE8545 BIOS 2.7.3, ConnectX-6 FW 20.33.1048
OS/Kernel	Ubuntu 20.04.4 LTS, GNU/Linux 5.4.0-190-generic x86_64
Drivers	NVIDIA GPU Driver 510.39.01, Mellanox OFED 5.6-1.0.3.3
Framework	PyTorch v1.12, CUDA 11.6, DeepSpeed 0.7.1

nodes is used to run the experiment. Both compute nodes are connected through an NVIDIA Spectrum SN3700 Ethernet switch supporting full-duplex 200 Gbps Ethernet with 12.8 Tbps switching capacity [122]. The cluster's topology is shown in Fig. 2-a. This small cluster represents generic or mainstream GPU clusters (e.g., TACC LoneStar6 [123]).

Since the basic form of Ethernet does not support Remote Direct Memory Access (RDMA), does not guarantee the arrival of traffic packets (lossy), and does not guarantee the latency of traffic packets [124, 125], we use RDMA over Converged Ethernet (RoCE) [126] in our setup. RoCE utilizes priority flow control to achieve lossless and guaranteed latency on traffic packets over commodity Ethernet with support for RDMA [127, 128]. RDMA and GPUDirect RDMA are vital for GPU clusters as they allow the Network Interface Cards (NICs) to directly transmit or receive the data from or to the CPU and GPU memory without the need to explicitly stage the data in the NIC's buffer, reducing the load of CPU and improving the latency [125, 129–132].

2) Compute node setup: Each XE8545 compute node has hardware and software configurations shown in Table II. The internal topology of XE8545 is shown in Fig. 2-b.

- **CPU:** Two AMD EPYC 7763 CPUs [133], each having 64 cores and 128 threads, are installed in each node. Each CPU

TABLE III
BANDWIDTH AND MEASUREMENT TOOLS

Interconnect	Interface	Links per node	Bandwidth[1]	Tools
CPU-DRAM	DRAM	8 × (2 CPUs)	25.6 GBps[2]	AMD μProf
CPU-CPU	xGMI	3	72 GBps[3]	AMD μProf
CPU-GPU	PCIe-GPU	1 × (4 GPUs)	64 GBps[4]	NVIDIA SMI
GPU-GPU	NVLink	12 × (4 GPUs)	50 GBps[5]	NVIDIA SMI
CPU-NIC	PCIe-NIC	1 × (2 NICs)	64 GBps[4]	AMD μProf
CPU-NVME	PCIe-NVME	1 × (8 NVMEs)	16 GBps[6]	AMD μProf
Internode	RoCE	1 × (2 NICs)	50 GBps[7]	HW Counter

[1] Theoretical bidirectional bandwidth per link (includes link overhead).
[2] Eight channels per CPU, half-duplex interface.
[3] Lane rate 18 GT/s, link width x16, bandwidth 36 GBps in each direction.
[4] Lane rate 16 GT/s, link width x16, bandwidth 32 GBps in each direction.
[5] Lane rate 50 GT/s, link width x4, bandwidth 25 GBps in each direction.
[6] Lane rate 16 GT/s, link width x4, bandwidth 8 GBps in each direction.
[7] 200 Gbps in each direction, two NICs per node.

has four NUMA domains (NPS4) for eight NUMA domains per compute node. Three cross-global memory interconnect links (xGMI) [134], also known as Infinity Fabric Inter-Socket (IFIS) [135], connect both CPUs.

- **Memory:** Each CPU has access to eight 64 GB DDR4-3200 ECC RDIMM memory in an eight-channel configuration to maximize the memory bandwidth. The total memory capacity for each compute node is 1024 GB.
- **GPU:** Four NVIDIA A100 SXM4 40 GB GPUs [136] are installed in each compute node. Each GPU is connected to the other three using four links of NVLink 3.0. In addition, a pair of GPUs are connected to each CPU through two PCIe 4.0 x16 links (link #1 and link #3).
- **NVME:** Two Intel D7-P5600 3.2 TB NVME SSDs [137] connected to CPU #1 are used as scratch disk and are configured in RAID0 using Linux `mdadm` [138]. In addition, a separate Intel D7-P5600 3.2 TB NVME SSD connected to CPU #0 is used for storing operating systems, libraries, and datasets. These SSDs are connected to their respective CPU using PCIe 4.0 x16 link (link #0) bifurcated to x4/x4/x4/x4.
- **NIC:** Two NVIDIA ConnectX-6 NICs [139] provide high-bandwidth communication links to other compute nodes. Each NIC is connected to each CPU using a PCIe 4.0 x16 link (link #2) and is configured to run in Ethernet mode.

B. Measurements

1) Application-level characterization: We perform application-level characterization to understand the system behavior better, particularly the bandwidth utilization. Using NVIDIA Nsight Systems (`nsys`) [140], we observe application behavior during each training step, including the kernels executed on GPUs, CUDA API calls, kernel/system calls, and resource utilization. We run each training configuration for 10 iterations and start collecting the data of the fifth iteration to give the system a chance to warm up.

2) Achieved model size: The GPT-2-like model is configured to have 16 attention heads, 2048 hidden sizes, 256 sequence lengths, and 1024 maximum position embeddings. We vary the number of layers of the model to change its size until it reaches the maximum size that particular hardware/software configurations can handle to run the training in mixed precision (FP16). The dataset used in the training is Wikipedia dump extracted using WikiExtractor [141] as recommended by Google Research [142]. The total parameters reported by DeepSpeed are used to represent the model size. The per-GPU batch size is 16 for all configurations.

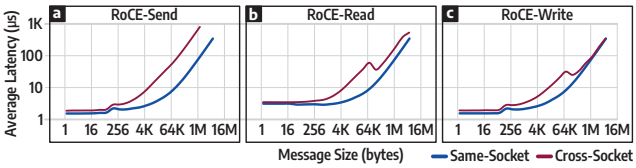

Fig. 3. The average latency of RoCE for different message sizes for channel semantic `SEND` (a), and memory semantic `RDMA READ` (b) and `RDMA WRITE` (c). Note that both axes are in log scale. Same-socket and cross-socket scenarios are drawn in blue and red, respectively.

3) Compute throughput: Compute throughput during the training is measured using the DeepSpeed Flops Profiler [120].

4) Memory usage: The CPU memory usage is measured using the Linux command `free` while the GPU memory usage is measured using `nvidia-smi` [143]. The NVME usage is measured using Linux command `df`.

5) Bandwidth: Table III summarizes all interconnects whose bandwidth is measured during the experiment using AMD μProf v3.6 [144], `nvidia-smi` [143], or low-level hardware counter. Aggregate bidirectional bandwidth is reported for each interconnect (e.g., 256 GBps for all four links of PCIe-GPU or 2400 GBps for all 48 links of NVLink).

C. Inter-node Latency and Bandwidth Stress Test

The test is conducted using OFED Performance Test [145] to see the inter-node bandwidth and latency for CPU and GPUDirect RDMA over Converged Ethernet (CPU-RoCE and GPU-RoCE). Two cases are explored: same-socket and cross-socket. Same-socket means the NIC is connected directly to the CPU running the test kernel (e.g., CPU #0 uses NIC #0 on Fig. 2). On the other hand, cross-socket means the NIC is connected to the neighboring CPU (e.g., CPU #0 uses NIC #1 on Fig. 2), and thus, the data must flow through the xGMI links. The test kernel is run in bidirectional mode and is pinned into the appropriate NUMA domain using `numactl` [146].

1) Latency test: Fig. 3 depicts the latency of RoCE for channel semantic `SEND` and memory semantic `RDMA READ` and `RDMA WRITE` with various message sizes. In summary, same-socket RoCE has latency under 6 μs while cross-socket RoCE has latency under 40 μs (i.e., seven times higher) for message size less than 64 kB.

2) CPU-RoCE bandwidth test: Four instances of the test kernel are run to stress the bandwidth, two in each CPU. Figure 4-a shows the bandwidth utilization during the test. In the same-socket scenario, the RoCE reaches 93% of the theoretical bandwidth (46 GBps out of 50 GBps). The PCIe #2, which connects the CPU to the NIC (Fig. 2-b), shows an average utilization of 48.88 GBps (76% of theoretical bandwidth). Four DRAM channels are observed to have higher traffic corresponding to which NUMA domain the kernel is run. Moving to the cross-socket scenario, RoCE only reaches 47% of the theoretical bandwidth (23.71 GBps out of 50 GBps) with some utilization on all xGMI links.

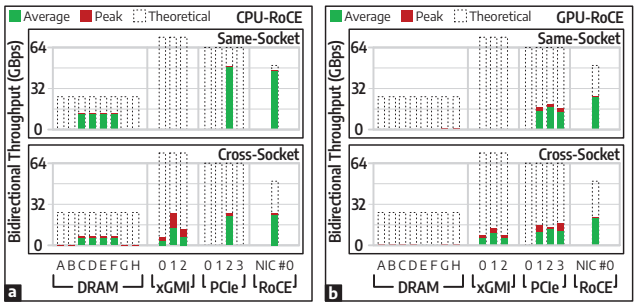

Fig. 4. The average, peak, and theoretical bandwidth of each interconnect during bandwidth stress-test for CPU-RoCE (a) and GPU-RoCE (b).

3) GPU-RoCE bandwidth test: Four instances of the test kernel are run to stress the bandwidth, one in each GPU. Figure 4-b shows the average and peak bandwidth during the test. Interestingly, both cases show unfavorable achieved bandwidth for RoCE: 52% and 42% of the theoretical bandwidth for same- and cross-socket cases, respectively. There is no significant traffic activity on DRAM since, by using GPUDirect RDMA, the NIC directly accesses GPU memory. Some traffic is observed on PCIe #1 and #3 in addition to PCIe #2 since they connect GPUs and NIC to their host CPU.

4) Hypothesis on bandwidth degradation: Unfavorable results for cross-socket CPU-RoCE, same-socket GPU-RoCE, and cross-socket GPU-RoCE need to be explored further, which necessitates looking into the I/O architecture of our CPU. The AMD EPYC 7763 CPU uses chiplet packaging technology consisting of eight Core Complex Dies (CCD) and one large I/O Dies (IOD) [147–149]. The IOD hosts eight channels of DDR4 memory controller and eight Global Memory Interconnect (GMI), also known as Infinity Fabric On Package (IFOP) [135, 147], to connect to eight CCDs [150]. It also hosts eight sets of x16 I/O Serializer/Deserializer (SerDes), 3 or 4 sets of which can be used as xGMI while the others are used for PCIe 4.0 x16 interfaces. The crossbar switch occupies the majority portion of the IOD middle area.

For the same-socket CPU-RoCE, traffic flows between the memory controller and the PCIe, which gives excellent results. For the other scenarios, traffic flows between two sets of SerDes (e.g., PCIe-PCIe, PCIe-xGMI, xGMI-xGMI), which shows unsatisfactory results. We hypothesize this degradation is due to the contention with the traffic routing inside the switch (Infinity Fabric Intra Die [148]) between two sets of I/O SerDes. However, AMD does not disclose the internal details of the switch nor give low-level access to measure traffic inside. Previous study shows concerns about the contention and scalability of centralized routing between chiplets [151].

IV. Evaluating DeepSpeed ZeRO

This section evaluates the advantages offered by DeepSpeed ZeRO at three stages (ZeRO-1, ZeRO-2, and ZeRO-3) by comparing them to the PyTorch Data Parallelism (DDP) [30] and Megatron-LM data and model parallelism (DP+MP) [31, 32]. For single-node training, the Megatron-LM is configured to have `TP=4` and `PP=4` while for dual-node training, it is configured to have `TP=8` and `PP=8`.

A. Application-level Characterization

1) Single Node: The first five timelines of Figure 5 show the execution characteristics of single training iteration for DDP, Megatron-LM, ZeRO-1, ZeRO-2, and ZeRO-3 when handling 1.4 billion parameters model on single node (four GPUs). The last four timelines will be discussed when ZeRO-Offload and ZeRO-Infinity are being used in Section V. The small timeline shows the zoom-in region of one forward phase (green line) and the beginning of the backward phase (dark-blue line). Megatron-LM, unlike the other configurations, has four pairs of forward and backward phases, which correspond to the number of model-parallel ranks being used (i.e., four GPUs). Each timeline has a different time scale, with DDP, Megatron-LM, ZeRO-1, ZeRO-2, and ZeRO-3 taking 471 ms, 736 ms, 412 ms, 404 ms, and 696 ms, respectively.

Common to all, general matrix-matrix multiplication (GEMM) is the majority of operations (green). These GEMM operations use Tensor Cores, available in NVIDIA GPUs, to accelerate computation. At the end of the forward phase, we observed element-wise operations (orange and pink). Weight update is done at the end of the backward phase (turqoise). Transform (red) and Memory (dark red) are memory-heavy.

We observe the use of NVIDIA Collective Communication Library (NCCL) to analyze the communication overhead. NCCL is used in all training configurations to perform reduce (lavender), broadcast (navy blue), all-gather (violet), and all-reduce (purple) operations. In theory, NCCL will try to detect node topology and use the fastest and shortest path for communication between NVIDIA GPUs (e.g., PCIe, NVLink, InfiniBand/Ethernet through NIC). DDP performs synchronization using All-Reduce at the end of the backward phase. In contrast, Megatron-LM performs significantly higher communication using All-Reduce between GEMM operations. ZeRO-1 and ZeRO-2 look similar to DDP, except ZeRO-2 uses Reduce during the backward phase. ZeRO-3 has more communication with All-Gather being used in between GEMM.

2) Dual Nodes: The application characteristic for dual-node training is similar to single-node training, except the time needed for NCCL communication is longer since it needs to use inter-node communication interfaces (NICs), which is the weakest link in our setup.

B. Achieved Model Size

1) Single Node: Fig. 6-a shows the achieved model size for single-node training. DDP is severely limited by the maximum memory of single GPU and can only fit a model with 1.4 billion parameters. On the other hand, Megatron-LM uses model parallelism to distribute the model across GPUs, allowing it to fit model with 5.5 billion parameters (i.e., almost four times larger than the DDP). ZeRO-1, which partitions the optimizer, can fit model with 4.4 billion parameters (i.e., 20% smaller than Megatron-LM). ZeRO-2, which further partitions the gradient, achieves model size of 5.2 billion parameters, slightly below the Megatron-LM. Finally, ZeRO-3, which partitions all model states, can handle model with 6.6 billion parameters (i.e., 20% larger than Megatron-LM).

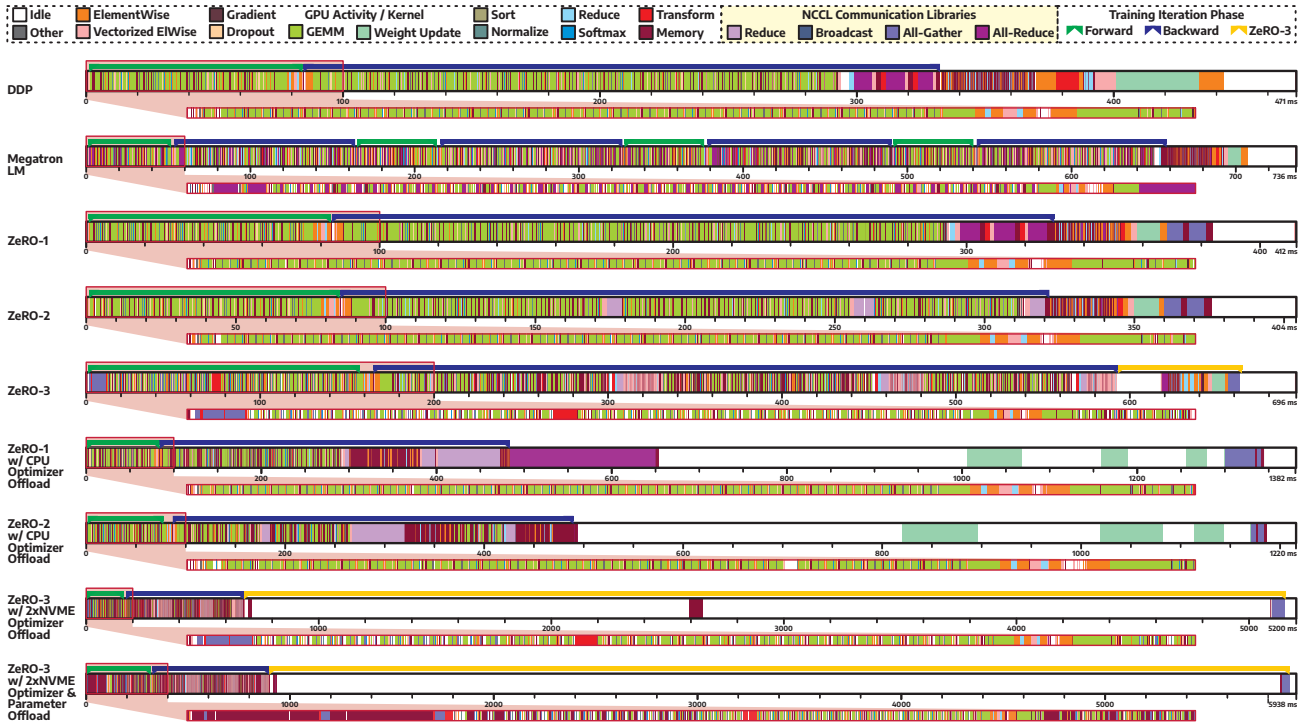

Fig. 5. Characterization of single iteration to train 1.4 billion parameters model on single node (four GPUs) for DDP (471 ms), Megatron-LM (736 ms), ZeRO-1 (412 ms), ZeRO-2 (404 ms), ZeRO-3 (696 ms), ZeRO-1 with CPU optimizer offload (1.38 s), ZeRO-2 with CPU optimizer offload (1.22 s), ZeRO-3 with 2 × NVME optimizer offload (5.2 s), and ZeRO-3 with 2 × NVME optimizer and parameter offload (5.9 s). Most kernels are GEMM; Megatron-LM, ZeRO-3, ZeRo-1 with offload, ZeRO-2 with offload, and ZeRO-3 with offload involve many NCCL communication kernels. Kernel execution may be overlapped using multiple CUDA streams. ZeRO-Offload and ZeRO-Infinity (Section V) should only be used for larger models that cannot fit without offloading them.

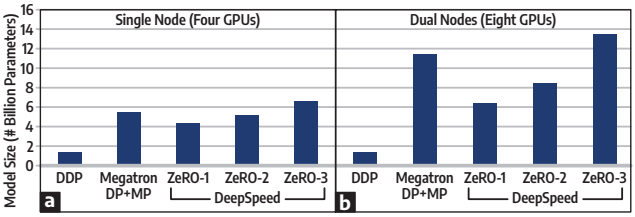

Fig. 6. Achieved model size for each configuration in single- (a) and dual-node (b) training. DeepSpeed ZeRO-3 can fit a larger model than the Megatron-LM while the ZeRO-2 falls slightly behind.

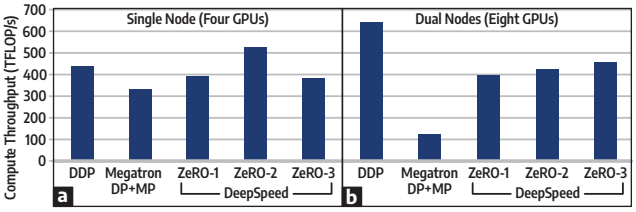

Fig. 7. Compute throughput for each configuration in single- (a) and dual-node (b) training. DeepSpeed has higher compute throughput than the Megatron-LM, with significant advantages observed on dual-node training.

2) Dual Nodes: Fig 6-b illustrates the achieved model size for dual-node training. The DDP can only fit model with 1.4 billion parameters despite the additional GPUs. The Megatron-LM can fit model with 11.4 billion parameters across two nodes (eight times larger than DDP). ZeRO-1 and ZeRO-2 can fit model with 6.4 and 8.5 billion parameters, respectively, across two nodes, which are still smaller than what Megatron-LM can achieve. Finally, ZeRO-3 can handle model with 13.5 billion parameters across two nodes, which is almost 20% larger than Megatron-LM.

C. Compute Throughput

1) Single Node: Fig. 7-a shows the achieved compute throughput for single-node training. The DDP achieved 438 TFLOP/s while Megatron-LM saw compute throughput drop to 331 TFLOP/s due to the higher communication overhead, which will be explained in Section IV-E1. ZeRO-1 achieves 391 TFLOP/s, 18.13% higher than Megatron-LM while still being able to handle 80% of the model size. Further partitioning the gradient in ZeRO-2 brings the throughput up significantly to 524 TFLOP/s, which is 58.3% larger than the Megatron-LM while being able to fit comparable model sizes. However, ZeRO-3 sees the compute throughput dropping to 381 TFLOP/s due to the higher communication from partitioning the model parameters. Nevertheless, it is still 15.1% higher in throughput while handling a 20% larger model compared to Megatron-LM. Fig. 8-a shows the trade-off between achieved model size and compute throughput for single-node training. Unless fitting larger model is needed, the ZeRO-2 is the sweet-spot, providing higher compute throughput while fitting a model size comparable to Megatron-LM.

2) Dual Nodes: Fig. 7-b shows the achieved compute throughput for dual-node training. DDP achieved 640 TFLOP/s, which is only a 46% increase from the single-

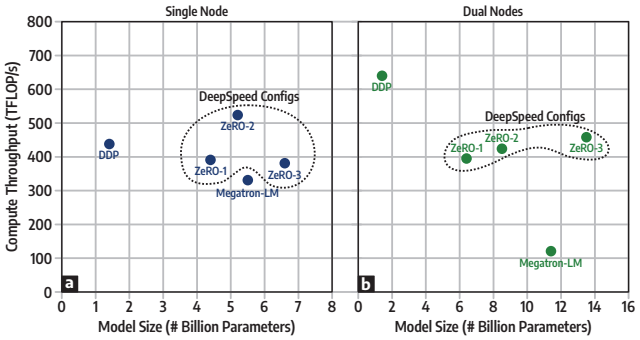

Fig. 8. The trade-off of compute throughput vs. achieved model sizes on single- (a) and dual-node (b) training. ZeRO-1, ZeRO-2, and ZeRO-3 are the various stages of DeepSpeed that help handle larger model sizes while maintaining high throughput. In single-node training, the Megatron-LM is fairly competitive with DeepSpeed. However, DeepSpeed achieves significantly higher throughput in dual-node training than Megatron-LM.

node training due to inter-node communication overhead. This overhead badly hurts Megatron-LM, which only achieved 121 TFLOP/s (81% lower than DDP). Section IV-E2 discusses this communication overhead in detail. Meanwhile, ZeRO-1, ZeRO-2, and ZeRO-3 achieved higher throughput than Megatron-LM: 395 TFLOP/s, 424 TFLOP/s, and 458 TFLOP/s, respectively. Interestingly, unlike Megatron-LM, DeepSpeed can maintain throughput when moving from single- to dual-node training while handling larger model sizes. ZeRO-1 in dual-node training achieves similar throughput as in single-node training while handling 45% larger model size. In addition, ZeRO-3 in dual-node training achieves 20% higher throughput than in single-node training while having double the model size. Fig. 8-b shows that ZeRO-3 is the best for maximizing model size while maintaining throughput.

D. Memory Usage

Since all configurations are aimed to hold the largest model possible on GPU memory, they have similar memory usage and memory composition: 154 GB to 157 GB GPU memory usage and 18 GB to 25 GB CPU memory usage, which may not be interesting to discuss further. However, the discussion on memory usage and memory composition becomes more interesting when we explore the offloading capabilities of DeepSpeed ZeRO-Offload and ZeRO-Infinity in Section V.

E. Bandwidth Utilization

1) Single Node: The first part of Table IV shows the average, 90th percentile, and peak bandwidth utilization for single-node training. With all model states fit into GPU memory, we observe minimal utilization of DRAM, xGMI, and PCIe bandwidth. All configurations have average DRAM bandwidth utilization under 1% of theoretical aggregate bandwidth (25.6 GBps x 16) across two CPUs. In addition, there is negligible inter-socket communication activity, with less than 1 GBps average utilization across all configurations. Furthermore, all four PCIe links that connect four GPUs to their respective CPUs have minimal usage.

Unlike other communication links, the NVLink does the most heavy lifting for handling inter-GPU communication.

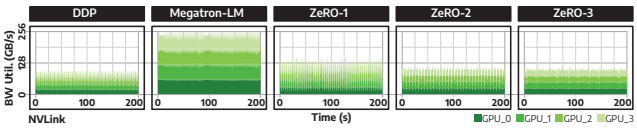

Fig. 9. NVLink bandwidth utilization pattern for single-node training.

Fig. 9 shows the utilization pattern of NVLink within 200 seconds. DDP has the lowest NVLink utilization compared to other configurations, with average and peak utilization at 83 GBps and 94.8 GBps, respectively. Since DDP replicates the model across all GPUs, lower communication is needed between the GPUs, mainly to synchronize the gradients. On the other hand, Megatron-LM has the highest average NVLink utilization, almost 300% higher than DDP, with peak utilization as high as 267 GBps. While it is still far from the theoretical aggregate bandwidth of NVLink (600GBps x 4), it may become a concern in dual-node training. Moving to DeepSpeed, ZeRO-1 has an average NVLink utilization of 111 GBps with a peak as high as 147 GBps. By further partitioning the gradients, ZeRO-2 reduces the average NVLink utilization to 97.3 GBps, peaking at 117 GBps. However, further partitioning the model parameters requires a slight increase in bandwidth, with an average of 99.7 GBps, peaking at 121 GBps. All DeepSpeed configurations show less NVLink utilization than Megatron-LM and are on par with DDP, in-line with application-level characterization (Section IV-A1).

2) Dual Nodes: Inter-node communication, often the weakest interconnect in latency and bandwidth, plays an important role [152] in multi-node training. We observe increased utilization in PCIe and RoCE while slightly less utilization in NVLink, as shown in the second part of Table IV. The PCIe bridges the GPUs and NICs through the CPUs, while the RoCE bridges the communication between the two nodes, and thus, higher utilization on these interconnects is expected.

Furthermore, minimal DRAM bandwidth utilization is observed since the GPUDirect RDMA is being used, However, we observe significant activities in xGMI links, with the highest average utilization at 10.4 GBps. This is 1100% higher than the highest average utilization in single-node. This indicates higher cross-socket traffic; a portion of inter-node traffic from the GPUs goes through the NIC connected to the neighboring CPU. This type of traffic is costly and has higher latency; thus, minimizing cross-socket traffic is essential.

Fig, 10 shows the bandwidth utilization pattern for NVLink, PCIe, and RoCE within 200 seconds. Like in the single-node training, DDP has the lowest bandwidth utilization with an average utilization of 9.28 GBps for RoCE and 60.2 GBps for NVLink. This low utilization, especially for RoCE, helps DDP achieve the highest compute throughput for dual-node training. Megatron-LM, which has been observed to have the highest NVLink utilization in single-node training, got a significant impact from using the significantly weaker inter-node link; its compute throughput dropped significantly. It has an average NVLink and RoCE utilization of 88.3 GBps and 13.8 GBps, respectively. The PCIe-GPU and PCIe-NIC have an average utilization of 16.9 GBps and 9.06 GBps, respectively. Looking

TABLE IV
BANDWIDTH UTILIZATION MEASUREMENT DATA (AVERAGE, 90TH PERCENTILE, PEAK)

Configuration	Aggregate Bidirectional Per-Node Bandwidth Utilization (GBps)																				
	DRAM			xGMI			PCIe									NVLink			RoCE		
							GPU			NVME			NIC								
	Avg	90th	Peak	Avg	90th	Peak	Avg	90th	Peak	Avg	90th	Peak	Avg	90th	Peak	Avg	90th	Peak	Avg	90th	Peak
Single Node (Section IV-E1)																					
PyTorch DDP	1.56	2.33	3.31	0.23	0.77	0.96	0.61	1.86	3.16	0.00	0.00	0.00	0.00	0.00	0.00	83.0	94.8	94.8	0.00	0.00	0.00
Megatron-LM	3.52	4.32	5.08	0.18	0.20	0.33	2.01	2.72	2.82	0.00	0.00	0.00	0.00	0.00	0.00	241	261	267	0.00	0.00	0.00
ZeRO-1	1.86	3.73	5.64	0.94	2.75	5.56	6.36	15.1	16.6	0.00	0.00	0.00	0.00	0.00	0.00	111	147	147	0.00	0.00	0.00
ZeRO-2	1.99	3.11	9.99	0.42	0.79	3.67	1.03	2.89	7.53	0.00	0.00	0.00	0.00	0.00	0.00	97.3	117	117	0.00	0.00	0.00
ZeRO-3	2.69	3.33	7.72	0.37	0.54	2.85	1.56	2.44	6.22	0.00	0.00	0.00	0.00	0.00	0.00	99.7	109	121	0.00	0.00	0.00
Dual Nodes (Section IV-E2)																					
PyTorch DDP	2.08	4.51	5.50	5.22	9.63	15.6	11.2	31.5	50.1	0.00	0.00	0.00	6.07	12	18.1	60.2	63.2	63.2	9.28	10.7	10.7
Megatron-LM	2.88	3.69	6.21	7.29	7.56	7.70	16.9	17.5	18.2	0.00	0.00	0.00	9.06	9.36	9.60	88.3	91.3	95.8	13.8	14.3	14.4
ZeRO-1	2.79	5.70	8.81	6.35	11.9	20.2	18.2	38.4	62.9	0.00	0.00	0.00	6.64	12.4	22.6	52.7	96.9	107	10.5	16.7	19.8
ZeRO-2	1.73	2.82	5.61	6.11	12.3	16.9	15.8	27.9	32.4	0.00	0.00	0.00	7.08	12.5	17.8	34.3	49.8	58.2	10.5	15.5	16.9
ZeRO-3	3.86	7.04	10.4	10.4	14.2	16.3	20.5	27.3	30.9	0.00	0.00	0.00	10.9	14.0	15.6	52.2	58.8	61.9	16.3	18.5	19.7
Consolidate Dual Nodes to Single Node with ZeRO-Offload (CPU Optimizer Offload) (Section V-A)																					
ZeRO-2 (CPU)	73.1	157	191	18.1	29.8	41.8	16.4	30.8	47.8	0.00	0.00	0.00	0.00	0.00	0.00	40.8	127	127	0.00	0.00	0.00
ZeRO-3 (CPU)	67.8	162	215	10.3	25.2	38.6	12.9	20.5	42.3	0.00	0.00	0.00	0.00	0.00	0.00	31.0	57.2	123	0.00	0.00	0.00
Consolidate Dual Nodes to Single Node with ZeRO-Infinity (1 × NVME Offload) (Section V-B)																					
Optimizer	15.1	25.2	130	2.28	7.18	40.8	1.53	1.1	30.3	0.29	0.02	13.9	0.00	0.00	0.00	6.72	2.3	109	0.00	0.00	0.00
Optimizer & Parameter	10.6	19.1	98.0	3.20	6.60	22.7	1.86	8.0	28.9	0.48	2.02	11.8	0.00	0.00	0.00	3.78	0.00	54.8	0.00	0.00	0.00
Consolidate Dual Nodes to Single Node with ZeRO-Infinity (2 × NVME Offload) (Section V-B)																					
Optimizer	23.6	83.7	142	3.87	16.6	34.7	3.21	16.5	50.9	3.13	6.14	6.32	0.00	0.00	0.00	10.1	64.1	128	0.00	0.00	0.00
Optimizer & Parameter	15.9	32.1	94.1	3.93	10.3	33.2	3.30	16.9	31.6	4.87	12.2	12.6	0.00	0.00	0.00	7.19	46.7	63.5	0.00	0.00	0.00
Largest Model Size for Single Node with ZeRO-Offload and ZeRO-Infinity (Section V-C)																					
ZeRO-1 (CPU)	60.7	155	189	5.07	14.4	24.9	16.8	41.4	63.4	0.00	0.00	0.00	0.00	0.00	0.00	31.4	70.2	99.7	0.00	0.00	0.00
ZeRO-2 (CPU)	66.3	132	158	17.5	30.2	49.5	19.3	39.6	58.6	0.00	0.00	0.00	0.00	0.00	0.00	38.9	102	159	0.00	0.00	0.00
ZeRO-3 (2 × NVME)	26.9	90.3	167	7.63	32.9	71.0	3.90	23.1	30.1	6.50	11.9	12.6	0.00	0.00	0.00	11.1	77.2	128	0.00	0.00	0.00

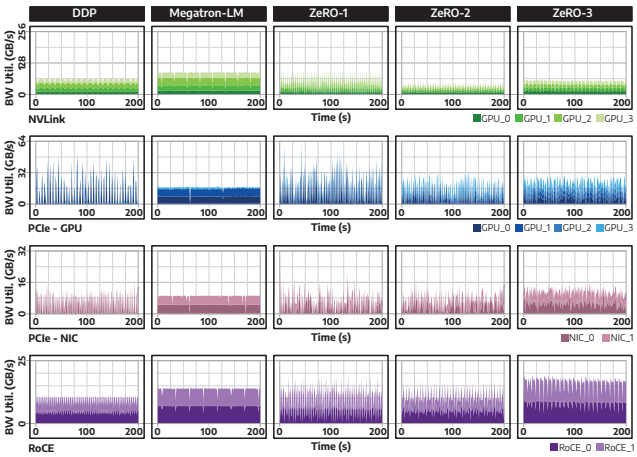

Fig. 10. From top to bottom: NVLink, PCIe-GPU, PCIe-NIC, and RoCE bandwidth utilization pattern for dual-node training.

at the pattern, more constant utilization is observed (shown in solid color for the entire time range) without significant peak value on all interconnects. This indicates abundant data needed to be transferred and, due to the hypothesized contention between I/O SerDes (Section III-C4), the transfer takes a long time to finish, which further explains the drop in throughput.

ZeRO-1 has average NVLink and RoCE utilization at 52.7 GBps and 10.5 GBps, with peaks as high as 107 GBps and 19.8 GBps, respectively. Instead of having a constant data transfer pattern, ZeRO-1 exhibits a peak-and-trough pattern. Although the peak utilization for ZeRO-1 can be 3.4x, 2.3x, and 1.4x higher than Megatron-LM for PCIe-GPU, PCIe-NIC, and RoCE, respectively, this type of data transfer pattern is somehow less prone to the hypothesized contention between I/O SerDes. Note that the stress test in Section III-C4 subjected these interconnects to a constant data transfer pattern. Further partitioning the gradients on ZeRO-2 reduces the average NVLink bandwidth utilization to 34.3 GBps while the average RoCE bandwidth utilization remains the same. The bandwidth utilization pattern is similar to ZeRO-1, albeit with a lower peak. Finally, ZeRO-3, which partitions the model parameter, shows an increase in RoCE utilization by 55% compared to ZeRO-1 and ZeRO-2, in line with DeepSpeed claim [35].

V. CONSOLIDATING MULTI-NODE INTO SINGLE-NODE

While the previous section discusses the advantages of DeepSpeed ZeRO, this section explores the distinctive features of DeepSpeed: the ZeRO-Offload [36] and ZeRO-Infinity [37].

A. Dual Nodes into Single Node with CPU Offload

As discussed in Section IV-B2, Megatron-LM can fit a model with 11.4 billion parameters in dual-node training. Utilizing the CPU offload feature on ZeRO-Offload, we try to fit the same model size into single node. For starters, ZeRO-Offload with ZeRO-1 is still insufficient, and thus, we only explore ZeRO-Offload on ZeRO-2 and ZeRO-3.

1) Compute Throughput: With ZeRO-Offload on ZeRO-2, a model with 11.4 billion parameters can fit inside single node. This configuration achieved a compute throughput of 191 TFLOP/s, 57.8% higher than Megatron-LM on dual nodes, as shown in Fig. 11-a. Further partitioning and offloading

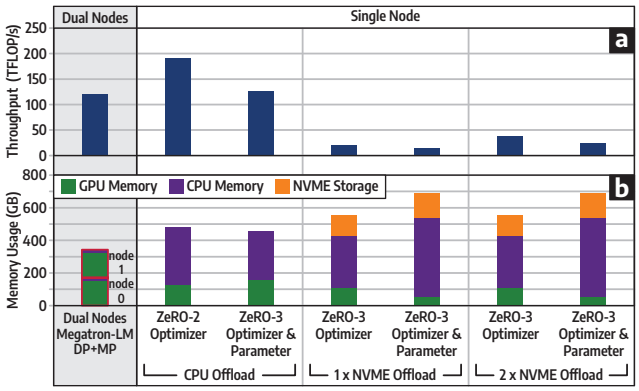

Fig. 11. Compute throughput (a) and memory usage (b) when using ZeRO-Offload or ZeRO-Infinity to consolidate dual-node into single-node training. The dual-node uses Megatron-LM to fit a model with a maximum 11.4 billion parameters. Single-node training with model states offloaded to CPU memory gives better throughput than dual-node with the same model size.

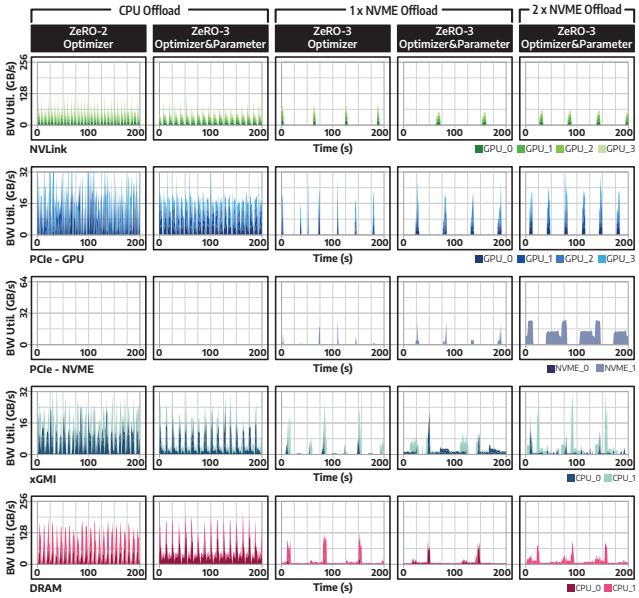

Fig. 12. From top to bottom: NVLink, PCIe-GPU, PCIe-NVME, xGMI, and DRAM bandwidth utilization pattern for single-node training with ZeRO-Offload (offloading to CPU) or ZeRO-Infinity (offloading to NVME storage).

model parameters using ZeRO-Offload on ZeRO-3 drops the compute throughput to 126 TFLOP/s since there is more data movement between CPU and GPU memory, adding more latency. However, it still gives comparable throughput to Megatron-LM. Unless fitting a larger model is necessary, it is recommended to use the ZeRO-Offload with ZeRO-2.

2) Memory Usage: Fig. 11-b shows the memory usage and composition for ZeRO-Offload on ZeRO-2 and ZeRO-3. The Megatron-LM uses 344 GB of memory across dual nodes, 308 GB (89.5%) of which is the GPU memory. On the other hand, ZeRO-Offload on ZeRO-2 uses 480 GB of memory, where 127 GB (26.5%) is on GPU and 353 GB (73.5%) is on CPU. The total memory used is 39.5% larger than the Megatron-LM on dual nodes because of the double buffers for overlapping communication and computation, hiding the data movement overhead. The ZeRO-Offload on ZeRO-3 has similar memory usage with a total of 452 GB, consisting of 157 GB GPU memory and 295 GB CPU memory.

Fine-tuning the size of buffers is needed to help DeepSpeed hide the data movement overhead based on the model size, the percentage of model states that are being offloaded, and the platform bandwidth availability (e.g., DRAM and PCIe between CPU and GPU). If the buffer is too large, the amount of usable space for storing the model states on GPU memory is significantly reduced, leading to more data movement. However, if the buffer is too small, it will reduce DeepSpeed's ability to overlap computation and communication.

3) Bandwidth Utilization: The third section of Table IV shows the bandwidth utilization for ZeRO-Offload with ZeRO-2 and ZeRO-3, while Fig. 12 shows their utilization pattern. With parts of model states being offloaded to CPU memory, it is expected to see higher utilization of DRAM bandwidth; ZeRO-Offload on ZeRO-2 has an average of 73.1 GBps with peak as high as 191 GBps while ZeRO-Offload on ZeRO-3 has an average of 67.8 GBps with peak as high as 215 GBps. The inter-socket xGMI also shows some utilization, with an average of 18.1 GBps and 10.3 GBps for ZeRO-2 and ZeRO-3, respectively. This indicates that the offloading mechanism may not take into account the topology of the platform, and thus, the GPU may need to access the offloaded model states stored in neighboring CPU memory. Finally, bandwidth utilization shows peak-and-trough patterns on PCIe-GPU, xGMI, and DRAM with minimal utilization on NVLink.

This observation aligns with the sixth and seventh timelines in Figure 5, which show significantly more All-Reduce and Reduce operations. The idle GPU time (white) is the communication overhead that cannot be overlapped since the model used to obtain the timeline in Figure 5 is too small (1.4 billion parameters) to justify CPU offload, resulting in the majority of time spent in data movement between CPU and GPU. Finally, it is worth noting that, during the idle time of the GPUs, the CPU is busy computing the optimizers stored in CPU memory.

B. Offloading to NVME Storage

We also explore the ZeRO-Infinity, which allows offloading model states to NVME storage. The ZeRO-Infinity is only available with ZeRO-3 and gives options to offload either the optimizer states or both optimizer states and model parameters. We use the same model with 11.4 billion parameters.

1) Compute Throughput: Fig. 11-a shows the compute throughput when NVME offload is enabled. Offloading optimizer states into single NVME drive achieves a throughput of 20.4 TFLOP/s. This throughput decreases when optimizer states and model parameters are offloaded to single NVME drive, achieving only 15.8 TFLOP/s. Adding a second NVME drive in RAID0 helps improve the throughput: 38.1 TFLOP/s (86.7% increase) for optimizer states and 24.5 TFLOP/s (55% increase) for optimizer states and model parameters.

2) Memory Usage: Fig. 11-b shows the memory usage and composition when NVME offloading is used. The total memory used for offloading optimizer states to single NVME

drive is 554 GB, with 108 GB (19.5%) on GPU, 317 GB on CPU (57.2%), and 129GB (23.2%) on NVME drive. The total memory used is 15.4% larger than offloading to CPU memory using ZeRO-Offload on ZeRO-2. This extra memory is used to implement double buffering to hide the data movement overhead: buffers in GPU memory, CPU memory, and NVME storage. Furthermore, offloading all model states to NVME storage increases the memory usage to 690 GB, with 52 GB (7.5%) on GPU, 488 GB (70.7%) on CPU, and 150 GB (21.7%) on NVME drive. GPU memory is less utilized since parts of model parameters reside in CPU memory and NVME storage. Generally, the more model parameters are held in GPU memory, the less data movement overhead between GPU, CPU, and NVME storage, yielding higher throughput. However, the free space on GPU memory can also be used for larger batch sizes, which may improve the throughput.

Adding a second NVME drive in RAID0 does not change the memory usage and composition since the RAID0 volume created using `mdadm` is transparent to DeepSpeed. However, this transparency may cause difficulty with data placement, especially when the NVME drives are connected to different CPUs, putting more strain on xGMI links and giving higher latency (Section V-E). Optimal data placement mainly depends on the topology and configuration of the compute nodes.

3) Bandwidth Utilization: The 4th and 5th part of Table IV shows the average and peak bandwidth utilization while Fig. 12 shows the utilization pattern for NVME offloading. For single NVME drive, the average bandwidth is 0.29 GBps and 0.48 GBps for offloading optimizer states and for offloading optimizer states and model parameters, respectively. However, we observe peak utilization as high as 13.9 GBps and 11.8 GBps. The abrupt peak and low utilization in PCIe-NVME links may relate to how the DRAM cache inside the NVME drive works. Reading and writing to NAND cells are significantly slower than DRAM, and thus, usually, NVME drive features a DRAM cache that acts as a buffer for reading and writing to NAND cells [153–155]. However, the DRAM cache has limited capacity, and when it is full or when the requested data is not cached, the speed drops dramatically.

Adding a second NVME drive helps fulfill the required bandwidth, where the average utilization bandwidth of PCIe-NVME climbs to 3.13 GBps and 4.87 GBps, which are 10.79x and 10.14x higher than the single NVME drive. This translates to almost double the compute throughput. Finally, the eighth and ninth timelines in Figure 5 show application-level characterization for NVME offloading. While the GPUs are idle (white), the CPU is busy computing the optimizers and exchanging data between NVME and CPU memory.

C. Largest Possible Model Size for Single Node

We also explored the largest model that can fit inside single node using ZeRO-Offload and ZeRO-Infinity as shown in Fig. 13. The ZeRO-Offload on ZeRO-1 can fit an 8.9 billion parameters model (i.e., 61.8% larger than Megatron-LM on single node). This configuration achieves 155.3 TFLOP/s of compute throughput and uses 229 GB of memory with 161 GB

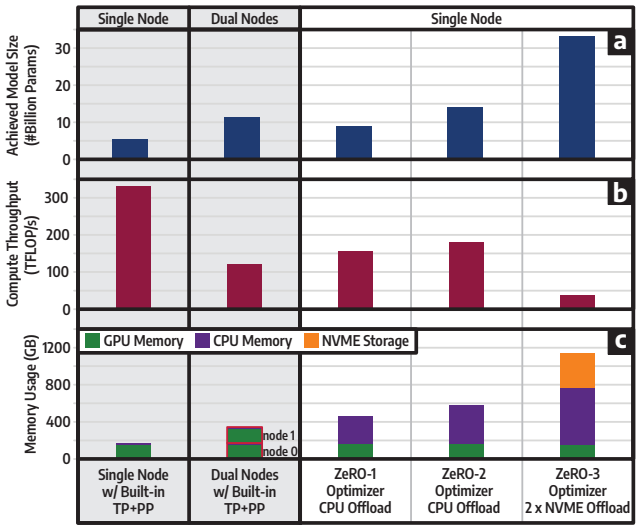

Fig. 13. Achieved model size (a), compute throughput (b), and memory usage (c) of DeepSpeed ZeRO-Offload and ZeRO-Infinity when handling the largest possible model on single-node training.

on GPU and 297 GB on CPU, Next, ZeRO-Offload on ZeRO-2 can fit the 14.2 billion parameters model. This is almost three times the size of the model that Megatron-LM can handle on a single node. This configuration achieves a compute throughput of 180.2 TFLOP/s while consuming 577 GB of memory with 158 GB on GPU and 419 GB on CPU. Finally, the ZeRO-Infinity on ZeRO-3 can fit a 33.3 billion parameters model, which is six times larger than Megatron-LM can fit on single node. This configuration offloads part of the model states to CPU memory and dual NVME drives, consuming a staggering 1144 GB of memory (158 GB on GPU, 611 GB on CPU, 375 GB on NVME drives). Due to the bottleneck in NVME storage bandwidth, it only achieves 37.16 TFLOP/s of compute throughput, which needs further investigation (Section V-E).

D. Sensitivity of Throughput to Model Size

Table V shows the sensitivity of throughput to the model size. In general, the throughput increases for bigger model sizes since there is more local work for each GPU and allows for better overlap between communication and computation. ZeRO-1 sees a drop in throughput at the maximum possible model size due to less memory available for double-buffering, while ZeRO-3 does not have a clear trend compared to others. In addition, both ZeRO-Offload (ZeRO-1 and ZeRO-2 with CPU Optimizer Offload) and ZeRO-Infinity (ZeRo-3 with 2 × NVME Optimizer Offload) have relatively stable throughput throughout various model sizes.

E. Optimizing NVME Data Placement

Before ending this section, we further explore data placement optimization for ZeRO-Infinity NVME offloading to handle the 33.3 billion parameters model. We added two additional NVME drives into CPU #0 in Figure 2 and created seven configurations shown in Figure 14. Configuration A and B have been explored in Section V-B and V-C. Configuration

TABLE V
SENSITIVITY OF THROUGHPUT TO MODEL SIZE (# BILLION PARAMETERS)

Config. \ Model Size	Throughput (TFLOP/s)														
	0.7	1.4	2.9	4.4	5.2	5.5	6.0	6.6	7.8	8.9	11.6	14.2	20.6	26.9	33.3
DDP	379	438													
Megatron-LM	270	309	312	315	324	331									
ZeRO-1	419	461	487	391											
ZeRO-2	427	472	502	509	524										
ZeRO-3	377	392	385	389	379	385	382	381							
ZeRO-1 (CPU)	145	165	148	167	150	168	164	163	158	155					
ZeRO-2 (CPU)	164	177	191	179	182	182	192	182	192	192	174	180			
ZeRO-3 (2 × NVME)	39	37	39	38	38	38	38	38	37	38	36	36	36	34	37

TABLE VI
ZERO-INFINITY VS. NVME CONFIGURATIONS

Config-uration	Through-put (TFLOP/s)	Aggr. Bidir. Bandwidth (GBps)					
		xGMI			PCIe-NVME		
		Avg	90th	Peak	Avg	90th	Peak
A	19.6	2.94	5.01	74.4	3.23	6.16	6.41
B	37.16	7.63	32.9	71.0	6.5	11.9	12.6
C	35.43	8.14	41.4	75.3	6.18	12.1	12.7
D	40.22	4.89	15.2	52.2	6.98	12.7	12.9
E	51.22	9.58	26.6	84.5	7.1	10.8	13.5
F	64.61	7.35	17.6	65.7	11.2	19.5	21.8
G	65.16	7.81	25.6	69.2	11.4	21.1	22.4

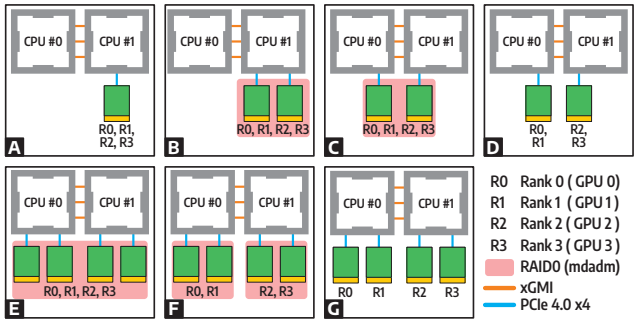

Fig. 14. Various configurations on how NVME drives are connected to the CPUs. Note that A and B have been explored in Section V-B and V-C.

D and G do not use RAID0, while F uses two RAID0 volumes, each with two disks. Since ZeRO-Infinity only supports one disk volume for offloading, we use the UNIX soft link to use multiple volumes by mapping each rank (0-3) to the appropriate disk/RAID0 volume, as shown in Figure 14. Each rank corresponds to the GPU index, and appropriate mapping of each rank to disk/RAID0 volume should consider the node topology. In addition, we performed parameter sweeps to configure DeepSpeed's asynchronous I/O optimally.

Table VI shows the achieved throughput and bandwidth utilization of xGMI and PCIe-NVME. As discussed in Section V-B, going from single NVME (Configuration A) to dual NVME (Configuration B) gave more than 80% increase in throughput. Configuration C still uses dual NVME in RAID0, but each CPU has direct access to one NVME drive. However, the achieved throughput is almost 5% lower, and average xGMI utilization is 7% more than Configuration B. Configuration D without RAID achieved the highest throughput for dual NVME configuration with the lowest xGMI utilization.

A similar phenomenon is observed with quad NVME configurations. Configuration E with a single RAID0 volume only achieved a 27.3% improvement in throughput compared to configuration D with dual NVME due to excessive data movement in xGMI. Configuration F with two RAID0 volumes and G without RAID0 achieved more than 60% higher throughput than the best dual NVME configuration thanks to lower xGMI utilization. Therefore, in this case, it is not recommended to have a RAID0 volume comprised of disks that span across different CPUs due to higher xGMI utilization.

Finally, we recommend that all available NVME slots be populated with fast NVME drives to get higher throughput when using ZeRO-Infinity. Only four out of eight available slots are populated with NVME drives for this experiment. If all eight slots are populated, the throughput will potentially be comparable to CPU offload. However, further data placement optimization is necessary to obtain higher throughput and lower data movement overhead depending on node topology.

VI. CONCLUSION

We explored the advantages of Microsoft DeepSpeed compared to PyTorch data parallelism (DDP) and Megatron-LM in distributed training of the GPT-2-like model. To do so, we built a small cluster that resembles a mainstream GPU cluster accessible to more researchers. We conducted bandwidth stress tests and hypothesized the contention between two sets of I/O Serializer-Deserializer (SerDes) inside the I/O die of AMD CPU degraded the attained bandwidth by around 50%.

Further, we compared DeepSpeed ZeRO with DDP and Megatron-LM. While DDP achieved higher compute throughput, it can only handle model size limited to single GPU memory capacity. Megatron-LM can fit 4x larger model than DDP while utilizing 300% more bandwidth. ZeRO can fit model with 0.8x-1.2x Megatron-LM size with less bandwidth. Both Megatron-LM and ZeRO are fairly competitive in single-node training. However, Megatron-LM saw significant drops in throughput on dual-node training due to excessive communication. ZeRO can fit a model with 0.56x-1.18x Megatron-LM size while giving 3.26x-3.78x higher throughput.

Finally, we use ZeRO-Offload to consolidate two nodes into single node, achieving 57.8% higher compute throughput than Megatron-LM on dual nodes. Utilizing NVME offloading in ZeRO-Infinity, we can fit model six times larger than what Megatron-LM can handle on single node. Adding more NVME drives will help with aggregate bandwidth to maintain throughput, and optimum data placement remains challenging, mainly depending on the compute node's topology.

ACKNOWLEDGMENT

This research was partly supported by National Science Foundation (NSF) Grant #2326894. Any opinions, findings, conclusions, or recommendations are those of the authors and not the funding agencies. The authors would like to express their sincere gratitude to Stephen Rousset, Liz Raymond, and Dennis Norwood of Dell Technologies for their support and assistance during this work. The authors would also like to thank Alex Rico of Advanced Micro Devices (AMD) for being the shepherd and the anonymous reviewers for their constructive feedback and suggestions.

REFERENCES

[1] T. Henighan, J. Kaplan, M. Katz, M. Chen, C. Hesse, J. Jackson, H. Jun, T. B. Brown, P. Dhariwal, S. Gray, C. Hallacy, B. Mann, A. Radford, A. Ramesh, N. Ryder, D. M. Ziegler, J. Schulman, D. Amodei, and S. McCandlish, "Scaling laws for autoregressive generative modeling," Oct 2020. [Online]. Available: http://arxiv.org/abs/2010.14701

[2] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre, "Training compute-optimal large language models," 2022.

[3] E. Caballero, K. Gupta, I. Rish, and D. Krueger, "Broken neural scaling laws," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=sckjveqlCZ

[4] K. Jing and J. Xu, "A survey on neural network language models," 2019.

[5] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," 2020.

[6] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[7] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. M. A. Patwary, Y. Yang, and Y. Zhou, "Deep learning scaling is predictable, empirically," 2017.

[8] J. Hestness, N. Ardalani, and G. Diamos, "Beyond human-level accuracy: Computational challenges in deep learning," in *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1–14. [Online]. Available: https://doi.org/10.1145/3293883.3295710

[9] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, "A comprehensive overview of large language models," 2023.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[11] W. Jeon, G. Ko, J. Lee, H. Lee, D. Ha, and W. W. Ro, "Chapter six - deep learning with gpus," in *Hardware Accelerator Systems for Artificial Intelligence and Machine Learning*, ser. Advances in Computers, S. Kim and G. C. Deka, Eds. Elsevier, 2021, vol. 122, pp. 167–215. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065245820300905

[12] W. G. Hatcher and W. Yu, "A survey of deep learning: Platforms, applications and emerging research trends," *IEEE Access*, vol. 6, pp. 24 411–24 432, 2018.

[13] S. Mittal and S. Vaishay, "A survey of techniques for optimizing deep learning on gpus," *Journal of Systems Architecture*, vol. 99, p. 101635, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762119302656

[14] M. Wang, W. Fu, X. He, S. Hao, and X. Wu, "A survey on large-scale machine learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 6, pp. 2574–2594, 2022.

[15] E. Nurvitadhi, G. Venkatesh, J. Sim, D. Marr, R. Huang, J. Ong Gee Hock, Y. T. Liew, K. Srivatsan, D. Moss, S. Subhaschandra, and G. Boudoukh, "Can fpgas beat gpus in accelerating next-generation deep neural networks?" in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 5–14. [Online]. Available: https://doi.org/10.1145/3020078.3021740

[16] B. Pourghassemi, C. Zhang, J. H. Lee, and A. Chandramowlishwaran, "On the limits of parallelizing convolutional neural networks on gpus," in *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*, ser. SPAA '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 567–569. [Online]. Available: https://doi.org/10.1145/3350755.3400266

[17] NVIDIA Corporation, "NVIDIA Tesla V100 GPU Architecture: The World's Most Advanced Data Center GPU," NVIDIA Corporation, California, US, Whitepaper, Aug. 2017. [Online]. Available: https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf

[18] ——, "NVIDIA A100 Tensor Core GPU Architecture: Unprecedented Acceleration at Every Scale," NVIDIA Corporation, California, US, Whitepaper, Jan. 2020. [Online]. Available: https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf

[19] D. Ganguli, D. Hernandez, L. Lovitt, A. Askell, Y. Bai, A. Chen, T. Conerly, N. Dassarma, D. Drain, N. Elhage, S. El Showk, S. Fort, Z. Hatfield-Dodds, T. Henighan, S. Johnston, A. Jones, N. Joseph, J. Kernian, S. Kravec, B. Mann, N. Nanda, K. Ndousse, C. Olsson, D. Amodei, T. Brown, J. Kaplan, S. McCandlish, C. Olah, D. Amodei, and J. Clark, "Predictability and surprise in large generative models," in *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1747–1764. [Online]. Available: https://doi.org/10.1145/3531146.3533229

[20] E. Ferrara, "Genai against humanity: Nefarious applications of generative artificial intelligence and large language models," 2023.

[21] M. Jovanović and M. Campbell, "Generative artificial intelligence: Trends and prospects," *Computer*, vol. 55, no. 10, pp. 107–112, 2022.

[22] A. Bandi, P. V. S. R. Adapa, and Y. E. V. P. K. Kuchi, "The power of generative ai: A review of requirements, models, input-output formats, evaluation metrics, and challenges," *Future Internet*, vol. 15, no. 8, 2023. [Online]. Available: https://www.mdpi.com/1999-5903/15/8/260

[23] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, and L. Sun, "A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt," 2023.

[24] O. Sharir, B. Peleg, and Y. Shoham, "The cost of training nlp models: A concise overview," 2020.

[25] K. CRAWFORD, *The Atlas of AI*. Yale University Press, 2023/12/15/ 2021. [Online]. Available: https://doi.org/10.2307/j.ctv1ghv45t

[26] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," 2019.

[27] A. E. Brownlee, J. Adair, S. O. Haraldsson, and J. Jabbo, "Exploring the accuracy – energy trade-off in machine learning," in *2021 IEEE/ACM International Workshop on Genetic Improvement (GI)*, 2021, pp. 11–18.

[28] L. F. W. Anthony, B. Kanding, and R. Selvan, "Carbontracker: Tracking and predicting the carbon footprint of training deep learning models," 2020.

[29] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, "Carbon emissions and large neural network training," 2021.

[30] S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania, and S. Chintala, "Pytorch distributed: Experiences on accelerating data parallel training," *Proc. VLDB Endow.*, vol. 13, no. 12, p. 3005–3018, aug 2020. [Online]. Available: https://doi.org/10.14778/3415478.3415530

[31] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," 2020.

[32] D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro, A. Phanishayee, and M. Zaharia, "Efficient large-scale language model training on gpu clusters using megatron-lm," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3458817.3476209

[33] V. Korthikanti, J. Casper, S. Lym, L. McAfee, M. Andersch, M. Shoeybi, and B. Catanzaro, "Reducing activation recomputation in large transformer models," 2022.

[34] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He, "Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 3505–3506. [Online]. Available: https://doi.org/10.1145/3394486.3406703

[35] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "Zero: Memory optimizations toward training trillion parameter models," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '20. IEEE Press, 2020.

[36] J. Ren, S. Rajbhandari, R. Y. Aminabadi, O. Ruwase, S. Yang, M. Zhang, D. Li, and Y. He, "ZeRO-Offload: Democratizing Billion-Scale model training," in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, Jul. 2021, pp. 551–564. [Online]. Available: https://www.usenix.org/conference/atc21/presentation/ren-jie

[37] S. Rajbhandari, O. Ruwase, J. Rasley, S. Smith, and Y. He, "Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3458817.3476205

[38] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, J. Chen, J. Chen, Z. Chen, M. Chrzanowski, A. Coates, G. Diamos, K. Ding, N. Du, E. Elsen, J. Engel, W. Fang, L. Fan, C. Fougner, L. Gao, C. Gong, A. Hannun, T. Han, L. Johannes, B. Jiang, C. Ju, B. Jun, P. LeGresley, L. Lin, J. Liu, Y. Liu, W. Li, X. Li, D. Ma, S. Narang, A. Ng, S. Ozair, Y. Peng, R. Prenger, S. Qian, Z. Quan, J. Raiman, V. Rao, S. Satheesh, D. Seetapun, S. Sengupta, K. Srinet, A. Sriram, H. Tang, L. Tang, C. Wang, J. Wang, K. Wang, Y. Wang, Z. Wang, Z. Wang, S. Wu, L. Wei, B. Xiao, W. Xie, Y. Xie, D. Yogatama, B. Yuan, J. Zhan, and Z. Zhu, "Deep speech 2 : End-to-end speech recognition in english and mandarin," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 173–182. [Online]. Available: https://proceedings.mlr.press/v48/amodei16.html

[39] M. Banko and E. Brill, "Scaling to very very large corpora for natural language disambiguation," in *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ser. ACL '01. USA: Association for Computational Linguistics, 2001, p. 26–33. [Online]. Available: https://doi.org/10.3115/1073012.1073017

[40] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 843–852.

[41] T. Linjordet and K. Balog, "Impact of training dataset size on neural answer selection models," in *Advances in Information Retrieval*, L. Azzopardi, B. Stein, N. Fuhr, P. Mayr, C. Hauff, and D. Hiemstra, Eds. Cham: Springer International Publishing, 2019, pp. 828–835.

[42] S. Uchida, S. Ide, B. K. Iwana, and A. Zhu, "A further step to perfect accuracy by training cnn with larger data," in *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016, pp. 405–410.

[43] D. Justus, J. Brennan, S. Bonner, and A. S. McGough, "Predicting the computational cost of deep learning models," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 3873–3882.

[44] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=B1xsqj09Fm

[45] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 1, jan 2020.

[46] P. Villalobos, J. Sevilla, T. Besiroglu, L. Heim, A. Ho, and M. Hobbhahn, "Machine learning model sizes and the parameter gap," 2022.

[47] N. Ardalani, C.-J. Wu, Z. Chen, B. Bhushanam, and A. Aziz, "Understanding scaling laws for recommendation models," 2022.

[48] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, nov 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[49] N. Pappas and T. Meyer, "A survey on language modeling using neural networks," *Infoscience: EPFL Scientific Publications*, 2012. [Online]. Available: http://infoscience.epfl.ch/record/192566

[50] M. U. Hadi, q. a. tashi, R. Qureshi, A. Shah, a. muneer, M. Irfan, A. Zafar, M. B. Shaikh, N. Akhtar, J. Wu, and S. Mirjalili, "Large language models: A comprehensive survey of its applications, challenges, limitations, and future prospects," *TechRxiv e-Prints*, Nov. 2023. [Online]. Available: http://dx.doi.org/10.36227/techrxiv.23589741.v4

[51] C. Ebert and P. Louridas, "Generative ai for software practitioners," *IEEE Software*, vol. 40, no. 4, pp. 30–38, 2023.

[52] L. Fan, L. Li, Z. Ma, S. Lee, H. Yu, and L. Hemphill, "A bibliometric review of large language models research from 2017 to 2023," 2023.

[53] J. Wu, W. Gan, Z. Chen, S. Wan, and P. S. Yu, "Multimodal large language models: A survey," 2023.

[54] S. Pahune and M. Chandrasekharan, "Several categories of large language models (llms): A short survey," *International Journal for Research in Applied Science and Engineering Technology*, vol. 11, no. 7, p. 615–633, Jul. 2023. [Online]. Available: http://dx.doi.org/10.22214/ijraset.2023.54677

[55] A. J. Thirunavukarasu, D. S. J. Ting, K. Elangovan, L. Gutierrez, T. F. Tan, and D. S. W. Ting, "Large language models in medicine," *Nature Medicine*, vol. 29, no. 8, pp. 1930–1940, Aug 2023. [Online]. Available: https://doi.org/10.1038/s41591-023-02448-8

[56] X. Yang, A. Chen, N. PourNejatian, H. C. Shin, K. E. Smith, C. Parisien, C. Compas, C. Martin, A. B. Costa, M. G. Flores, Y. Zhang, T. Magoc, C. A. Harle, G. Lipori, D. A. Mitchell, W. R. Hogan, E. A. Shenkman, J. Bian, and Y. Wu, "A large language model for electronic health records," *npj Digital Medicine*, vol. 5, no. 1, p. 194, Dec 2022. [Online]. Available: https://doi.org/10.1038/s41746-022-00742-2

[57] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "BioBERT: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, Feb. 2020.

[58] K. He, R. Mao, Q. Lin, Y. Ruan, X. Lan, M. Feng, and E. Cambria, "A survey of large language models for healthcare: from data, technology, and applications to accountability and ethics," 2023.

[59] Y. Yang, M. C. S. UY, and A. Huang, "Finbert: A pretrained language model for financial communications," 2020.

[60] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann, "Bloomberggpt: A large language model for finance," 2023.

[61] I. Beltagy, K. Lo, and A. Cohan, "Scibert: A pretrained language model for scientific text," 2019.

[62] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, and R. Stojnic, "Galactica: A large language model for science," 2022.

[63] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. M. Zhang, "Large language models for software engineering: Survey and open problems," 2023.

[64] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models," in *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3491101.3519665

[65] S. Zhang, Z. Chen, Y. Shen, M. Ding, J. B. Tenenbaum, and C. Gan, "Planning with large language models for code generation," 2023.

[66] L. Belzner, T. Gabor, and M. Wirsing, "Large language model assisted software engineering: Prospects, challenges, and a case study," in *Bridging the Gap Between AI and Reality*, B. Steffen, Ed. Cham: Springer Nature Switzerland, 2024, pp. 355–374.

[67] F. Zeng, W. Gan, Y. Wang, N. Liu, and P. S. Yu, "Large language models for robotics: A survey," 2023.

[68] C. Zhang, J. Chen, J. Li, Y. Peng, and Z. Mao, "Large language models for human–robot interaction: A review," *Biomimetic Intelligence and Robotics*, vol. 3, no. 4, p. 100131, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667379723000451

[69] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 523–11 530.

[70] J. Huang and K. C.-C. Chang, "Towards reasoning in large language models: A survey," 2023.

[71] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, R. Zheng, X. Fan, X. Wang, L. Xiong, Y. Zhou, W. Wang, C. Jiang, Y. Zou, X. Liu, Z. Yin, S. Dou, R. Weng, W. Cheng, Q. Zhang, W. Qin, Y. Zheng, X. Qiu, X. Huang, and T. Gui, "The rise and potential of large language model based agents: A survey," 2023.

[72] Y. Zhu, H. Yuan, S. Wang, J. Liu, W. Liu, C. Deng, Z. Dou, and J.-R. Wen, "Large language models for information retrieval: A survey," 2023.

[73] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin, and M. Du, "Explainability for large language models: A survey," 2023.

[74] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237. [Online]. Available: https://aclanthology.org/N18-1202

[75] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS'20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[76] R. Gozalo-Brizuela and E. C. Garrido-Merchan, "Chatgpt is not all you need. a state of the art review of large generative ai models," 2023.

[77] T. Wu, S. He, J. Liu, S. Sun, K. Liu, Q.-L. Han, and Y. Tang, "A brief overview of chatgpt: The history, status quo and potential future development," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 5, pp. 1122–1136, 2023.

[78] P. P. Ray, "Chatgpt: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 121–154, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S266734522300024X

[79] D. Patel and G. Wong, "Gpt-4 architecture, infrastructure, training dataset, costs, vision, moe," https://www.semianalysis.com/p/gpt-4-architecture-infrastructure?utm_source=%2Fsearch%2FGPT-4&utm_medium=reader2, 2023.

[80] B. Hanindhito and L. K. John, "Accelerating ml workloads using gpu tensor cores: The good, the bad, and the ugly," in *Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3629526.3653835

[81] NVIDIA Corporation, "NVIDIA H100 Tensor Core GPU Architecture: Exceptional Performance, Scalability, and Security for The Data Center," NVIDIA Corporation, California, US, Whitepaper, Jan. 2022. [Online]. Available: https://resources.nvidia.com/en-us-tensor-core/gtc22-whitepaper-hopper

[82] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, E. Zhang, R. Child, R. Y. Aminabadi, J. Bernauer, X. Song, M. Shoeybi, Y. He, M. Houston, S. Tiwary, and B. Catanzaro, "Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model," 2022.

[83] J. Sevilla, L. Heim, A. Ho, T. Besiroglu, M. Hobbhahn, and P. Villalobos, "Compute trends across three eras of machine learning," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1109/IJCNN55064.2022.9891914

[84] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen, "A survey of large language models," 2023.

[85] M. Isaev, N. McDonald, and R. Vuduc, "Scaling infrastructure to support multi-trillion parameter LLM training," in *Architecture and System Support for Transformer Models (ASSYST @ISCA 2023)*, 2023. [Online]. Available: https://openreview.net/forum?id=rqn2v1Ltgn0

[86] M. Houston, "Nvidia selene: Leadership-class supercomputing infrastructure," https://www.nvidia.com/en-us/on-demand/session/supercomputing2020-sc2019/, 2020.

[87] T. D. Le, T. Sekiyama, Y. Negishi, H. Imai, and K. Kawachiya, "Involving cpus into multi-gpu deep learning," in *Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 56–67. [Online]. Available: https://doi.org/10.1145/3184407.3184424

[88] Y. Jiang, Y. Zhu, C. Lan, B. Yi, Y. Cui, and C. Guo, "A unified architecture for accelerating distributed DNN training in heterogeneous GPU/CPU clusters," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, Nov. 2020, pp. 463–479. [Online]. Available: https://www.usenix.org/conference/osdi20/presentation/jiang

[89] O. Valery, P. Liu, and J.-J. Wu, "Cpu/gpu collaboration techniques for transfer learning on mobile devices," in *2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*, 2017, pp. 477–484.

[90] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Comput. Surv.*, vol. 53, no. 2, mar 2020. [Online]. Available: https://doi.org/10.1145/3377454

[91] M. Langer, Z. He, W. Rahayu, and Y. Xue, "Distributed training of deep learning models: A taxonomic perspective," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 12, pp. 2802–2818, 2020.

[92] Y. Ko, K. Choi, J. Seo, and S.-W. Kim, "An in-depth analysis of distributed training of deep neural networks," in *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2021, pp. 994–1003.

[93] S.-X. Zou, C.-Y. Chen, J.-L. Wu, C.-N. Chou, C.-C. Tsao, K.-C. Tung, T.-W. Lin, C.-L. Sung, and E. Y. Chang, "Distributed training large-scale deep architectures," in *Advanced Data Mining and Applications*, G. Cong, W.-C. Peng, W. E. Zhang, C. Li, and A. Sun, Eds. Cham: Springer International Publishing, 2017, pp. 18–32.

[94] Y. Kim, H. Choi, J. Lee, J.-S. Kim, H. Jei, and H. Roh, "Efficient large-scale deep learning framework for heterogeneous multi-gpu cluster," in *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2019, pp. 176–181.

[95] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu, "Petuum: A new platform for distributed machine learning on big data," *IEEE Transactions on Big Data*, vol. 1, no. 2, pp. 49–67, 2015.

[96] H. Li, A. Kadav, E. Kruus, and C. Ungureanu, "Malt: Distributed data-parallelism for existing ml applications," in *Proceedings of the Tenth European Conference on Computer Systems*, ser. EuroSys '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2741948.2741965

[97] V. Gupta, D. Choudhary, P. Tang, X. Wei, X. Wang, Y. Huang, A. Kejariwal, K. Ramchandran, and M. W. Mahoney, "Training recommender systems at scale: Communication-efficient model and data parallelism," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 2928–2936. [Online]. Available: https://doi.org/10.1145/3447548.3467080

[98] Y. Kim, J. Lee, J.-S. Kim, H. Jei, and H. Roh, "Efficient multi-gpu memory management for deep learning acceleration," in *2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2018, pp. 37–43.

[99] Y. Lecun, "A theoretical framework for back-propagation," in *Proceedings of the 1988 Connectionist Models Summer School, CMU, Pittsburg, PA*, D. Touretzky, G. Hinton, and T. Sejnowski, Eds. Morgan Kaufmann, 1988, pp. 21–28.

[100] M. Zinkevich, M. Weimer, L. Li, and A. Smola, "Parallelized stochastic gradient descent," in *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., vol. 23. Curran Associates, Inc., 2010. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2010/file/abea47ba24142ed16b7d8fbf2c740e0d-Paper.pdf

[101] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. a. Ranzato, A. Senior, P. Tucker, K. Yang, Q. Le, and A. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/6aca97005c68f1206823815f66102863-Paper.pdf

[102] J. Geng, D. Li, and S. Wang, "Elasticpipe: An efficient and dynamic model-parallel solution to dnn training," in *Proceedings of the 10th Workshop on Scientific Cloud Computing*, ser. ScienceCloud '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 5–9. [Online]. Available: https://doi.org/10.1145/3322795.3331463

[103] S. Pal, E. Ebrahimi, A. Zulfiqar, Y. Fu, V. Zhang, S. Migacz, D. Nellans, and P. Gupta, "Optimizing multi-gpu parallelization strategies for deep learning training," *IEEE Micro*, vol. 39, no. 5, pp. 91–101, 2019.

[104] A. Castelló, M. F. Dolz, E. S. Quintana-Ortí, and J. Duato, "Analysis of model parallelism for distributed neural networks," in *Proceedings of the 26th European MPI Users' Group Meeting*, ser. EuroMPI '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3343211.3343218

[105] Z. Jia, M. Zaharia, and A. Aiken, "Beyond data and model parallelism for deep neural networks." in *Proceedings of Machine Learning and Systems*, A. Talwalkar, V. Smith, and M. Zaharia, Eds., vol. 1, 2019, pp. 1–13. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2019/file/b422680f3db0986ddd7f8f126baaf0fa-Paper.pdf

[106] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, P. B. Gibbons, and M. Zaharia, "Pipedream: Generalized pipeline parallelism for dnn training," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, ser. SOSP '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1–15. [Online]. Available: https://doi.org/10.1145/3341301.3359646

[107] J. Tarnawski, A. Phanishayee, N. Devanur, D. Mahajan, and F. N. Paravecino, "Efficient algorithms for device placement of dnn graph operators," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS'20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[108] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, and z. Chen, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/093f65e080a295f8076b1c5722a46aa2-Paper.pdf

[109] I. Rocha, N. Morris, L. Y. Chen, P. Felber, R. Birke, and V. Schiavoni, "Pipetune: Pipeline parallelism of hyper and system parameters tuning for deep learning clusters," in *Proceedings of the 21st International Middleware Conference*, ser. Middleware '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 89–104. [Online]. Available: https://doi.org/10.1145/3423211.3425692

[110] N. Takisawa, S. Yazaki, and H. Ishihata, "Distributed deep learning of resnet50 and vgg16 with pipeline parallelism," in *2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW)*, 2020, pp. 130–136.

[111] D. Narayanan, A. Phanishayee, K. Shi, X. Chen, and M. Zaharia, "Memory-efficient pipeline-parallel dnn training," 2021.

[112] B. Wang, Q. Xu, Z. Bian, and Y. You, "Tesseract: Parallelize the tensor parallelism efficiently," in *Proceedings of the 51st International Conference on Parallel Processing*, ser. ICPP '22. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3545008.3545087

[113] W. Xu, Y. Zhang, and X. Tang, "Parallelizing dnn training on gpus: Challenges and opportunities," in *Companion Proceedings of the Web Conference 2021*, ser. WWW '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 174–178. [Online]. Available: https://doi.org/10.1145/3442442.3452055

[114] S. Li, H. Liu, Z. Bian, J. Fang, H. Huang, Y. Liu, B. Wang, and Y. You, "Colossal-ai: A unified deep learning system for large-scale parallel training," in *Proceedings of the 52nd International Conference on Parallel Processing*, ser. ICPP '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 766–775. [Online]. Available: https://doi.org/10.1145/3605573.3605613

[115] Q. Xu and Y. You, "An efficient 2d method for training super-large deep learning models," in *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2023, pp. 222–232. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/IPDPS54959.2023.00031

[116] HuggingFace, "How to use megatron-lm," https://huggingface.co/docs/accelerate/usage_guides/megatron_lm, 2023.

[117] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[118] HuggingFace, "How to use deepspeed," https://huggingface.co/docs/accelerate/usage_guides/deepspeed, 2023.

[119] Microsoft Corporation, "Deepspeed: Extreme-scale model training for everyone," https://www.microsoft.com/en-us/research/blog/deepspeed-extreme-scale-model-training-for-everyone/, 2020.

[120] ——, "Deepspeed flops profiler," https://www.deepspeed.ai/tutorials/flops-profiler/, 2023.

[121] Dell Inc., "Dell EMC PowerEdge XE8545 - Technical Specification," Dell Inc., Technical Specification, 2020. [Online]. Available: https://dl.dell.com/topicspdf/poweredge-xe8545_owners-manual2_en-us.pdf

[122] NVIDIA Corporation, "NVIDIA Spectrum SN3000 Series Switches - Data Center Performance, Scale, and Rich Telemetry," NVIDIA Corporation, Datasheet, 2021. [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/networking/sn3000-series-datasheet-1557133.pdf

[123] Texas Advanced Computing Center, "Lonestar6 user guide," https://docs.tacc.utexas.edu/hpc/lonestar6/, 2023.

[124] J. P. Kenny and C. D. Ulmer, "Roce: Promising technology for ethernet as a high performance networking fabric." Sandia National Lab. (SNL-CA), United States, Tech. Rep., 2019. [Online]. Available: https://doi.org/10.2172/1573446

[125] A. Tekin, A. Durak, C. Piechurski, D. Kaliszan, F. A. Sungur, F. Robertsén, and P. Gschwandtner, "State-of-the-art and trends for computing and interconnect network solutions for hpc and ai," *Partnership for Advanced Computing in Europe*, 2021.

[126] R. Mittal, A. Shpiner, A. Panda, E. Zahavi, A. Krishnamurthy, S. Ratnasamy, and S. Shenker, "Revisiting network support for rdma," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 313–326. [Online]. Available: https://doi.org/10.1145/3230543.3230557

[127] A. Shpiner, E. Zahavi, O. Dahley, A. Barnea, R. Damsker, G. Yekelis, M. Zus, E. Kuta, and D. Baram, "Roce rocks without pfc: Detailed evaluation," in *Proceedings of the Workshop on Kernel-Bypass Networks*, ser. KBNets '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 25–30. [Online]. Available: https://doi.org/10.1145/3098583.3098588

[128] J. P. Kenny, J. J. Wilke, C. D. Ulmer, G. M. Baker, S. Knight, and J. A. Friesen, "An evaluation of ethernet performance for scientific workloads," in *2020 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS)*, 2020, pp. 57–67.

[129] S. Potluri, K. Hamidouche, A. Venkatesh, D. Bureddy, and D. K. Panda, "Efficient inter-node mpi communication using gpudirect rdma for infiniband clusters with nvidia gpus," in *2013 42nd International Conference on Parallel Processing*, 2013, pp. 80–89.

[130] S. S. Sharkawi and G. A. Chochia, "Communication protocol optimization for enhanced gpu performance," *IBM Journal of Research and Development*, vol. 64, no. 3/4, pp. 9:1–9:9, 2020.

[131] A. Venkatesh, H. Subramoni, K. Hamidouche, and D. K. Panda, "A high performance broadcast design with hardware multicast and gpudirect rdma for streaming applications on infiniband clusters," in *2014 21st International Conference on High Performance Computing (HiPC)*, 2014, pp. 1–10.

[132] K. Hamidouche, A. Venkatesh, A. A. Awan, H. Subramoni, C.-H. Chu, and D. K. Panda, "Exploiting gpudirect rdma in designing high performance openshmem for nvidia gpu clusters," in *2015 IEEE International Conference on Cluster Computing*, 2015, pp. 78–87.

[133] Advanced Micro Devices, Inc., "AMD EPYC™ 7763," https://www.amd.com/en/products/cpu/amd-epyc-7763, 2021.

[134] M. Velten, R. Schöne, T. Ilsche, and D. Hackenberg, "Memory performance of amd epyc rome and intel cascade lake sp server processors," in *Proceedings of the 2022 ACM/SPEC on International Conference on Performance Engineering*, ser. ICPE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 165–175. [Online]. Available: https://doi.org/10.1145/3489525.3511689

[135] N. Beck, S. White, M. Paraschou, and S. Naffziger, "'zeppelin': An soc for multichip architectures," in *2018 IEEE International Solid-State Circuits Conference - (ISSCC)*, 2018, pp. 40–42.

[136] TechPowerUp, "Nvidia a100 sxm4 40 gb," https://www.techpowerup.com/gpu-specs/a100-sxm4-40-gb.c3506, 2020.

[137] Intel Corporation, "Intel® SSD D7-P5500 and Intel® SSD D7-P5600 Series Data Center, PCI Express (PCIe), 96-Layer TLC Intel ®3D NAND," Intel Corporation, Datasheet, 2020. [Online]. Available: https://www.solidigm.com/content/dam/solidigm/en/site/products/data-center/product-briefs/d7-p5500-p5600-product-brief/documents/d7-p5600-p5500-series-brief.pdf

[138] W. Reese, "Increase performance, reliability and capacity with software raid," *Linux J.*, vol. 2008, no. 175, nov 2008.

[139] NVIDIA Corporation, "NVIDIA ConnectX-6 DX Eth-

ernet SmartNIC," NVIDIA Corporation, Datasheet, 2022. [Online]. Available: https://nvdam.widen.net/s/qpszhmhpzt/networking-overal-dpu-datasheet-connectx-6-dx-smartnic-1991450
[140] NVIDIA. Corporation, "Nvidia nsight systems," https://developer.nvidia.com/nsight-systems, 2022.
[141] G. Attardi, "Wikiextractor," https://github.com/attardi/wikiextractor, 2015.
[142] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[143] NVIDIA. Corporation, "Nvidia system management interface," https://https://developer.nvidia.com/nvidia-system-management-interface/, 2020.
[144] Advanced Micro Devices, Inc., "AMD μProf," https://www.amd.com/en/developer/uprof.html, 2022.
[145] OpenFabrics Alliance, "Open fabrics enterprise distribution (ofed) performance tests," https://github.com/linux-rdma/perftest, 2023.
[146] C. Lameter, "Numa (non-uniform memory access): An overview: Numa becomes more common because memory controllers get close to execution units on microprocessors." *Queue*, vol. 11, no. 7, p. 40–51, jul 2013. [Online]. Available: https://doi.org/10.1145/2508834.2513149
[147] S. Naffziger, K. Lepak, M. Paraschou, and M. Subramony, "2.2 amd chiplet architecture for high-performance server and desktop products," in *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, 2020, pp. 44–45.
[148] D. Suggs, M. Subramony, and D. Bouvier, "The amd "zen 2" processor," *IEEE Micro*, vol. 40, no. 2, pp. 45–52, 2020.
[149] M. Mattioli, "Rome to milan, amd continues its tour of italy," *IEEE Micro*, vol. 41, no. 4, pp. 78–83, 2021.
[150] S. Atchley, C. Zimmer, J. Lange, D. Bernholdt, V. Melesse Vergara, T. Beck, M. Brim, R. Budiardja, S. Chandrasekaran, M. Eisenbach, T. Evans, M. Ezell, N. Frontiere, A. Georgiadou, J. Glenski, P. Grete, S. Hamilton, J. Holmen, A. Huebl, D. Jacobson, W. Joubert, K. Mcmahon, E. Merzari, S. Moore, A. Myers, S. Nichols, S. Oral, T. Papatheodore, D. Perez, D. M. Rogers, E. Schneider, J.-L. Vay, and P. K. Yeung, "Frontier: Exploring exascale," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3581784.3607089
[151] P. Fotouhi, S. Werner, J. Lowe-Power, and S. J. B. Yoo, "Enabling scalable chiplet-based uniform memory architectures with silicon photonics," in *Proceedings of the International Symposium on Memory Systems*, ser. MEMSYS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 222–334. [Online]. Available: https://doi.org/10.1145/3357526.3357564
[152] Z. Zhang, C. Chang, H. Lin, Y. Wang, R. Arora, and X. Jin, "Is network the bottleneck of distributed training?" in *Proceedings of the Workshop on Network Meets AI & ML*, ser. NetAI '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 8–13. [Online]. Available: https://doi.org/10.1145/3405671.3405810
[153] D. Takashima, M. Noguchi, N. Shibata, K. Kanda, H. Sukegawa, and S. Fujii, "An embedded dram technology for high-performance nand flash memories," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 2, pp. 536–546, 2012.
[154] H. Lin, Z. Sha, J. Li, Z. Cai, B. Gerofi, Y. Shi, and J. Liao, "Dram cache management with request granularity for nand-based ssds," in *Proceedings of the 51st International Conference on Parallel Processing*, ser. ICPP '22. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3545008.3545081
[155] H. Shim, B.-K. Seo, J.-S. Kim, and S. Maeng, "An adaptive partitioning scheme for dram-based cache in solid state drives," in *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, 2010, pp. 1–12.