Modeling Rack-Scale Data and ZeRO Parallelisms

Namo Samuthrsindh
paramuth@mit.edu

Prompt Kerdphoksup
promptkp@mit.edu

I. Introduction

Data parallelism increases the throughput of deep neural network (DNN) inference by storing a copy of the model on every GPU. Although this strategy is effective for small models, it is impossible to store the whole model on a single GPU as models increase in size. To address this issue, ZeRO parallelism [1] splits different parts of the model across multiple GPUs, reducing the memory bottlenecks.

However, the strategy comes with the cost of network communication between GPUs. Therefore, a hardware designer must answer the following question: How does the additional cost in terms of latency and energy scale with different aspects of the system such as the number of GPUs, the throughput of each GPU, the filter types, etc.

To answer this question, we model a rack-scale DNN system with multiple GPUs in Timeloop [2] and use Accelergy [3] to calculate the energy and latency. Although Timeloop does not natively support network modeling, we design a model to approximate the energy and latency costs of the network which differ based on network topologies. Apart from three supported topologies, a customized topology can be easily added by the user.

In summary, this work introduces a model for estimating the energy and latency costs of ZeRO parallelism compared to Data parallelism. Applying our model across diverse workloads and system architectures, we show that ZeRO incurs significantly lower overhead for workloads where model weights constitute a small portion of the total data, such as in convolutional layers. Additionally, the strategy scales more efficiently in terms of energy consumption when increasing the number of processing elements (PEs) per GPU, rather than the number of GPUs. Moreover, network topology can further influence performance. A fully connected topology yields the greatest improvements, but at the expense of increased complexity and implementation cost. Alternatively, transitioning from a linear to a ring topology offers substantial performance gains with minimal structural changes. Finally, our model is generalizable and can be applied by practitioners to analyze their own workloads and architectures, enabling informed decisions about parallelization strategies.

II. ZeRO Modeling Strategy

ZeRO parallelism works by splitting different weights across multiple GPUs. We denote the number of GPU by N, so each GPU stores $1/N$ of each layer. During inference, it has to fetch the rest of the missing weights from other GPUs over the network. After it finishes using a layer, it will retain $1/N$ of the layer and discard the other $(N-1)/N$ so that the space can be reused for the next layer.

Our goal is to calculate the latency and the energy for a given workload. We separate the calculation into two parts: within GPU, and across GPU.

A. Within GPU

This calculation includes the energy and the latency caused by data movements between memory hierarchies and multiply-accumulate (MAC) operations by the PEs, which will be handled by Timeloop. The main challenge is that ZeRO parallelism fetches some of the weights over the network, so some data movement may have higher latency and energy cost. Although Timeloop does not support this feature, we get around this challenge by assuming perfect latency hiding. Since the number of cycles which the computation with each layer takes can be precomputed, each GPU can efficiently prefetch the weights of the next layer, so this assumption is practical.

B. Across GPU

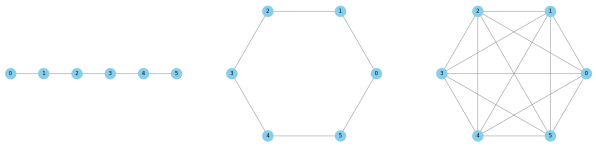

Fig. 1: Linear, ring, and fully connected network topologies

Despite perfect latency hiding, the data transferred over network cost additional energy. We assume that one network hop costs a constant amount of energy. However, sending a packet from one GPU to another may require multiple hops depending on the network topology. Therefore, given that the traffic distribution is uniform, we calculate the average number of network hops for each communication in each topology. Currently, we support linear, ring, and fully connected topologies (Figure 1).

Additionally, we consider the network bandwidth which we define as the maximum number of bytes that can be transferred in a cycle. Although the perfect latency hiding assumption is necessary for the ease of modeling in Timeloop, it only works if the computation is GPU-bound and not network-bound, i.e., when the amount of data transferred in a cycle exceeds the network bandwidth. Thus, we must compare the network data per cycle to the network bandwidth. The exact formula will be given in the next section.

III. Experiment Setup

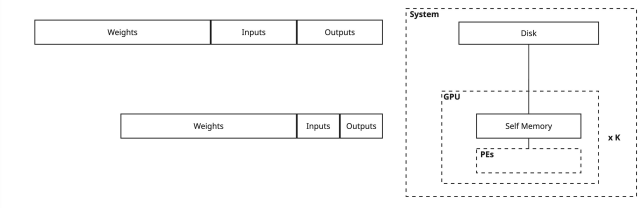

Fig. 2: Rack-scale architecture in Timeloop

Figure 2 shows how we model the system in Timeloop. The highest memory level is the disk which is the source of all information. The disk communicates with multiple GPUs which execute independently. Each GPU has its own SRAM and multiple PEs.

To run a workload, the input batch is spatially partitioned across the GPUs so that each GPU handles exactly one input. For data parallelism, the results from Timeloop are the final results. For ZeRO parallelism, the results are combined with the latency and energy costs from network communication across GPUs.

First, we calculate the total amount of data transferred across the network in bytes, denoted by B. As described in section 2, each GPU have to fetch $(N-1)/N$ of the layer over the network, so we have

$$B = \frac{N-1}{N} \times L \times N = (N-1) \times L$$

where L is the size of the layer in bytes.

A. Energy

The energy cost within GPU is equal for data parallel and ZeRO parallel, and we get this value directly from Timeloop. For the network cost, we use the formula

$$E_{network} = B \times AvgHops \times EnergyPerHopPerByte$$

where $AvgHops$ is the average number of hops each communication takes (calculation omitted for brevity), and $EnergyPerHopPerByte$ is the energy cost for sending a byte of data over one network hop. The latter varies based on the hardware, so it can be set to any values by the user. Currently, we use $1.6 \times 10^{-10} J$.

B. Latency

Additional latency from the network only occurs when the amount of data transferred per cycle is over the network bandwidth, which we also call the threshold. Denote the number of cycles given by Timeloop and the actual number of cycles by $Cycle_{Timeloop}$ and $Cycle_{actual}$ respectively. Then, if T is the maximum number of bytes per cycle, we have the formula

$$Cycle_{actual} = \max(Cycle_{Timeloop}, B \times AvgHops/T).$$

IV. Results and Discussion

The results indicate that ZeRO parallelism is most effective when the proportion of weights in the workload is relatively low, such as in convolutional layers. In contrast, for workloads involving substantial data transfer, ZeRO parallelism introduces significant energy overhead and offers limited scalability in terms of latency.

The scalability limitations of ZeRO parallelism are primarily constrained by the number of GPUs and the available network bandwidth. As the number of GPUs increases, the amount of weight sharing also increases, leading to a linear rise in energy overhead and network-bound computation. Furthermore, while increasing the number of PEs can reduce computational latency, it also exposes the bottleneck in network latency.

Finally, improvements in network topology, in terms of reduced average communication latency, can enhance the performance of ZeRO parallelism. This is because a more efficient topology reduces the average number of hops per communication, and thus, reduces the total amount of data in the network. However, such improvements may come at the cost of increased complexity in network design and scalability.

A. Number of GPUs

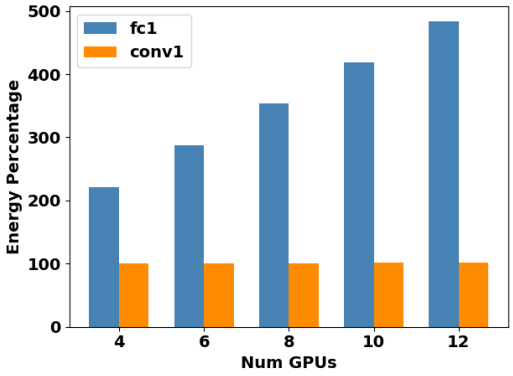

Fig. 3: Energy overhead of ZeRO parallelism relative to Data parallelism across varying GPU counts, using sixteen PEs and a linear network topology under both fully connected and convolutional workload

To assess scalability, we vary the number of GPUs and measure the energy and latency of ZeRO parallelism as a percentage relative to Data parallelism. In this setup, Data parallelism maintains consistent energy consumption and latency per GPU.

As illustrated in Figure 3, the energy overhead of ZeRO parallelism scales linearly with the number of GPUs. This trend arises from the linear increase in average energy consumed per byte of data transferred in a linear network topology. However, the rate of increase in total energy percentage depends on the workload. For weight-intensive tasks like fully connected layers, the energy growth is much steeper. In contrast, convolutional workloads exhibit minimal increase. As can be seen in Figure 3, with twelve GPUs, the fully connected workload's

energy percentage approaches 500%, while the convolutional workload remains just above 100%.

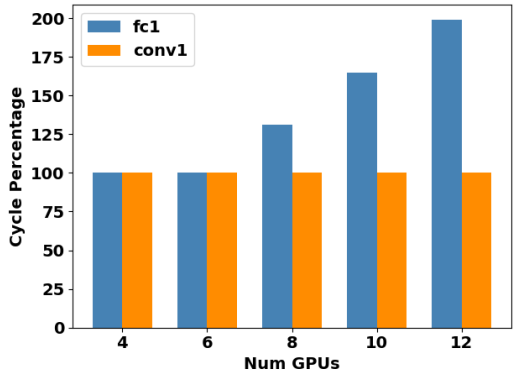

Fig. 4: Latency (in cycles) overhead of ZeRO parallelism relative to Data parallelism across varying GPU counts, using sixteen PEs and a linear network topology under both fully connected and convolutional workload

In terms of latency, Figure 4 shows that the convolutional workload achieves perfect scaling, with no overhead in the range of GPUs we have tested. This is because the communication cost for the convolutional layer's weight is small relative to its computation time, allowing effective latency hiding. However, for workloads with a higher proportion of parameters, such as fully connected tasks, latency hiding degrades as the number of GPUs increases. Beyond six GPUs, network communication becomes the bottleneck, and the latency overhead reaches 200% at twelve GPUs. This indicates that increasing the number of GPUs is only beneficial when the workload has a relatively low communication-to-computation ratio.

B. Number of Processing Elements

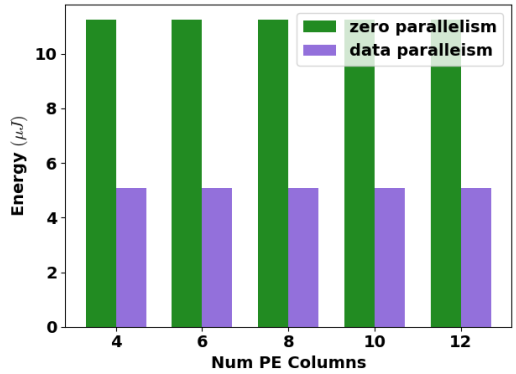

Fig. 5: Energy consumption (in μJ) for the fully connected workload using ZeRO and Data parallelism across varying numbers of PE columns, with four GPUs and a linear network topology

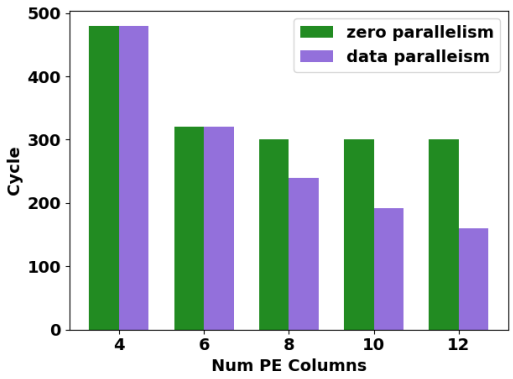

Fig. 6: Latency (in cycles) for the fully connected workload using ZeRO and Data parallelism across varying numbers of PE columns, with four GPUs and a linear network topology

Another potential architectural optimization involves using more powerful GPUs, which we model by increasing the number of PEs per GPU. Figure 5 presents the energy consumption of the fully connected workload under ZeRO and Data parallelism, with the number of GPUs kept constant.

As shown in the figure, ZeRO parallelism consistently results in higher energy consumption compared to Data parallelism. Increasing the number of PEs per GPU, however, has little to no impact on energy usage for either strategy. This behavior is expected because for each GPU, additional PEs improve computational latency but do not significantly affect the energy required for that computation. Moreover, the amount of data transferred remains unchanged regardless of PE count, and since data movement is the main contributor to energy overhead in ZeRO, scaling PEs does not increase energy consumption for communication. Therefore, from an energy perspective, increasing PEs is a practical and efficient optimization.

Latency presents a distinct challenge compared to energy consumption. As shown in Figure 6, communication latency in ZeRO parallelism remains hidden only up to six PEs. Beyond this point, the computation becomes network-bound, causing the overall latency to plateau at approximately 300 cycles. In contrast, Data parallelism continues to benefit from increased PEs, with latency gradually decreasing.

This behavior occurs because increasing the number of PEs enhances computational throughput, thereby reducing computation time. However, network latency remains unchanged since the amount of weight data transferred does not vary. Once the threshold for latency hiding is exceeded, the benefits of additional PEs are reached.

It is worth noting that, although increasing the number of PEs leads to similar latency behavior as adding more GPUs, it does not significantly impact energy consumption. In contrast, increasing the number of GPUs raises both latency and energy costs. Therefore, scaling up PEs is generally a more energy-efficient optimization. The main limitation of this approach lies in the hardware constraints of improving the capability of

each GPU.

C. Network Topology

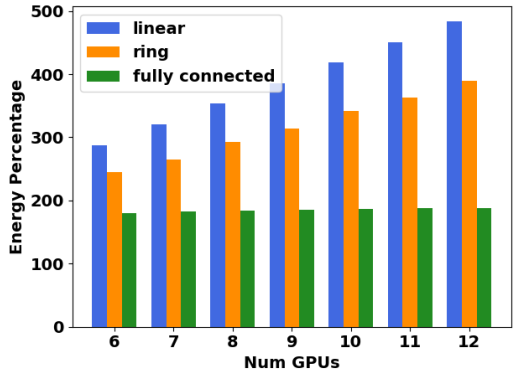

Fig. 7: Energy overhead of ZeRO parallelism relative to Data parallelism across varying GPU counts, evaluated using three network topologies—linear, ring, and fully connected—with sixteen PEs, under a fully connected workload

Since the primary bottleneck arises from data transfer, it is evident that optimizing the network topology can significantly reduce overhead. Figure 7 illustrates the energy overhead associated with a fully connected workload across different numbers of GPUs, using three network topologies: linear, ring, and fully connected. As shown, the fully connected topology results in minimal, nearly constant energy overhead regardless of GPU count. This is because data is transferred over a single hop, keeping energy costs consistent. While the ring topology also offers improvements over the linear configuration, it still exhibits a linearly increasing energy overhead as the number of GPUs grows.

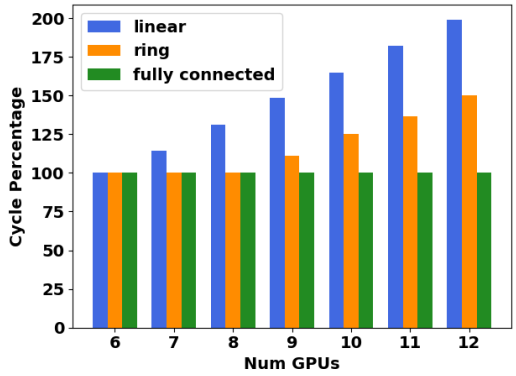

Fig. 8: Latency (in cycles) overhead of ZeRO parallelism relative to Data parallelism across varying GPU counts, evaluated using three network topologies—linear, ring, and fully connected—with sixteen PEs, under a fully connected workload

A similar pattern is observed in latency as well. Figure 8 demonstrates that the ring topology can mitigate latency bottlenecks better than the linear configuration. In a linear topology, communication latency becomes a limiting factor after six GPUs, whereas a ring topology extends this threshold to eight GPUs. In contrast, the fully connected topology effectively hides transfer latency altogether, as each data packet travels only one hop, thereby avoiding the linear growth in latency seen in the other topologies.

These results clearly indicate that a fully connected topology is ideal for scalable computation due to its consistent performance. However, implementing such a topology introduces challenges, including increased hardware complexity and higher costs due to the need for additional connections per GPU. Alternatively, the ring topology, which requires only one additional connection compared to the linear setup, offers a noticeable improvement in ZeRO-parallelism efficiency. Thus, in scenarios where cost and scalability to a large number of GPUs are critical, the ring topology may present a more practical trade-off.

V. Conclusion and Future Work

This work presents a model for evaluating the energy and latency costs of ZeRO parallelism in comparison to Data Parallelism. By applying the model to a range of workloads and hardware configurations, we show that ZeRO introduces minimal overhead when model weights make up a small portion of the overall workload, as seen in convolutional layers. The approach also scales more efficiently in terms of energy when increasing the number of PEs per GPU rather than the number of GPUs. While fully connected topologies offer the best performance, some modest changes, such as shifting from a linear to a ring topology, can yield meaningful gains with lower complexity.

In future work, to obtain more precise results, we plan to implement thread-pinning simulation, allowing specific data to be bound to designated memory locations. This enhancement would eliminate the need to assume uniform network latency and enable the use of Timeloop to explicitly model each network topology based on user-defined architectures.

References

[1] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "Zero: Memory optimizations toward training trillion parameter models," 2020.

[2] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: A systematic approach to dnn accelerator evaluation," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 304–315, 2019.

[3] Y. N. Wu, J. S. Emer, and V. Sze, "Accelergy: An architecture-level energy estimation methodology for accelerator designs," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2019.