Exploring Memory Expansion Designs for Training Mixture-of-Experts Models

Taekyung Heo[1], Saeed Rashidi[1, 3†], Changhai Man[1], Divya Kiran Kadiyala[1], William Won[1], Sudarshan Srinivasan[2], Midhilesh Elavazhagan[2], Madhu Kumar[2], Alexandros Daglis[1], Tushar Krishna[1]

[1]Georgia Institute of Technology [2]Intel [3]Hewlett Packard Enterprise

I. Introduction

Machine learning (ML) has achieved impressive success in a variety of fields [3], [6], [8], [15], [19]. It is widely recognized that increasing model parameters can enhance ML model capabilities [18]. As a result, the sizes of ML models have grown exponentially in recent years. However, this expansion in model parameters introduces computational and memory-related challenges. Mixture-of-Experts (MoE) has emerged as a potential solution to mitigate computational costs [13], [20], [26], [31]. In traditional dense models, the computational cost increases linearly with the model size since all parameters are involved in the training process. MoE models differ from traditional dense models as they partition and selectively activate a subset of parameters, resulting in improved model quality without substantially increasing computational costs. However, MoE models require additional memory capacity, leading to the adoption of high-capacity, expensive HBM for quick off-GPU memory access. As GPU memory has not scaled proportionately, it has given rise to the challenge known as the *GPU memory wall* [28]. Memory expansion techniques can help overcome the GPU memory wall challenge by enabling GPUs to access remote memory. Although memory expansion has been studied for decades, it has not been explored in the context of training MoE models. This study aims to investigate a range of memory expansion design options, with the goal of optimizing both performance and performance per cost for training 1 trillion parameter MoE models.

II. Background

Various techniques have been proposed to overcome the GPU memory wall.

Memory Optimization Techniques. ZeRO [27] is a representative memory optimization technique. ZeRO-DP optimizes the model states by partitioning model parameters and removing replicates while ZeRO-R optimizes the residual states. ZeRO-DP has three stages: ZeRO-1, ZeRO-2, and ZeRO-3. Fully-sharded data parallelism (FSDP) [5], [24], [36] is a library similar to ZeRO-3 that enables parameter sharding. Other previously proposed memory optimization techniques include memory-efficient optimizers [2], [32], recomputation [7], [25], [33], and memory harvesting [9].

Memory Offloading Techniques. The key idea behind memory offloading is to offload model parameters or optimizer states to remote memory, which is located away from local GPU memory. ZeRO-offload [29] alleviates the pressure on GPU memory by offloading optimizer states to CPU memory. ZeRO-Infinity allows GPUs to utilize CPU memory and NVMe memory in addition to their local HBM memory [28].

Memory Disaggregation for Distributed Training. While similar to memory offloading in terms of utilizing remote memory, memory disaggregation assumes a unified address space and seamless data access with cache-coherent networks. Memory disaggregation has been studied for several decades [10], [17], [21]. However, due to the absence of cache-coherent memory networks, earlier designs were closer to memory offloading relying on RDMA and paging mechanisms [1], [16], [22], [30]. CXL arises as a fundamental solution, enabling genuine memory disaggregation by providing memory expansion, a unified address space, coherency, and high performance.

III. Memory Expansion Design Options

In this study, we consider both memory offloading and memory disaggregation as memory expansion techniques. We evaluate the performance of four memory expansion designs inspired by current systems from CPU/GPU vendors, as illustrated in Fig. 1. The configurations for these design options

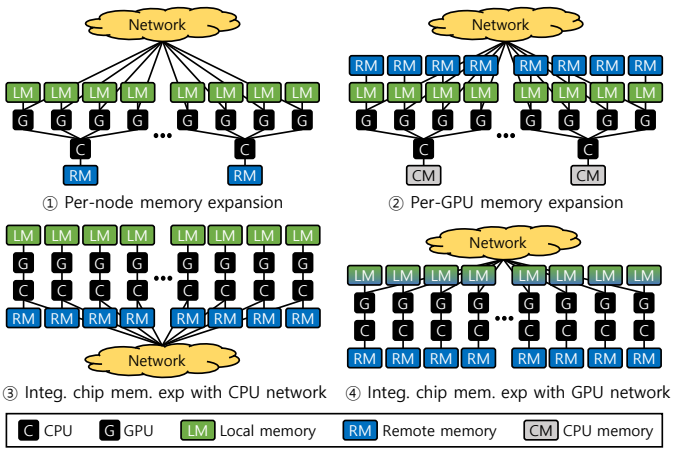

Fig. 1: Memory expansion design points.

	Peak Perf	Local Mem BW	Remote Mem BW	GPU Link BW
❶	2048 TFLOPS	4096 GB/sec	128 GB/sec	900 GB/sec
❷	2048 TFLOPS	4096 GB/sec	512 GB/sec	900 GB/sec
❸	2048 TFLOPS	4096 GB/sec	900 GB/sec	512 GB/sec
❹	2048 TFLOPS	4096 GB/sec	900 GB/sec	900 GB/sec

TABLE I: Memory expansion design point configurations.

†This work was done while the author was affiliated with Georgia Tech

Memory Offloading	Per-GPU Required HBM Capacity	Per-GPU Required Remote Memory Capacity
`Conservative`	16.3GB	72.9GB
`Aggressive`	6.25GB	82.9GB

TABLE II: Required memory capacity when training MoE.

are shown in Table I. To simulate future GPU architectures, we set the peak performance of GPUs to 2048 TFLOPS and local memory bandwidth to 4096 GB/s for all design options.

❶ **Per-node Expansion** [11], [14]. The system consists of CPU-GPU nodes, with each node comprising four GPUs connected to a CPU via PCIe. GPUs use the CPU-side memory as remote memory and a portion of the trained model can be offloaded there. The GPUs are interconnected through a high-bandwidth network, such as NVLinks.

❷ **Per-GPU Expansion** [4]. Similar to per-node expansion, but each GPU also has CXL-attached remote memory. GPUs do not access CPU memory or other GPUs' memory.

❸ & ❹ **Integrated-Chip Expansion** [12], [23]. This system includes an integrated chip comprising a CPU and a GPU, connected through a high-speed interconnect with a 900 GB/s bandwidth. The CPU-side memories can offer a total bandwidth of 1000 GB/s, accessible by the GPU via the chip-to-chip interconnect. To attain enhanced performance, multiple integrated chips can be interconnected. Network connections can be established using either CPUs (❸) or GPUs (❹).

IV. Methodology

Workloads. We evaluate MoE with two distinct memory offloading configurations. In particular, we utilize the DeepSpeed-MoE model [26] with 1T parameters with 256 GPUs. MoE is modeled in the form of directed acyclic graphs, where nodes represent memory access, computation, and communication, derived from real-world models.

Simulator. We use ASTRA-sim [35] to evaluate the four different architectures due to its capability to run training tasks while adjusting system design parameters. GPUs are modeled with a roofline model [34]. Local memory access is modeled as part of the roofline model, under the assumption that it is pipelined with computation, while remote memory access is modeled with a simple bandwidth model.

Memory Offloading Options. We evaluate two memory offloading options: conservative and aggressive. `Conservative` keeps weights and activation checkpoints in GPU HBM memory while pushing back and popping recomputed activations from remote memory. In contrast, `aggressive` retains weights, activation checkpoints, and recomputed activations in remote memory, pulling them into GPU HBM memory layer-by-layer and writing them back to remote memory after computation. `Aggressive` minimizes the required per-GPU HBM capacity while maintaining efficient memory management during training. Table II presents the per-GPU required memory capacity to train the MoE model with each of the offloading options.

V. Evaluation

Fig. 2 shows the normalized training time for the MoE model using four memory expansion designs and two memory

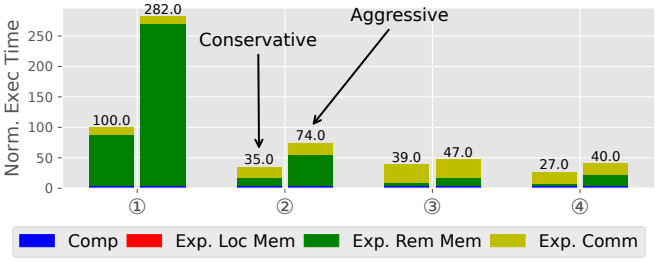

Fig. 2: Normalized execution time to train MoE.

offloading options. The execution time is normalized with respect to the per-node memory expansion (❶) with the `conservative` offloading. Each bar is divided into four segments: compute time, exposed local memory access time, exposed remote memory access time, and exposed communication time. We refer to any memory access or communication time that is not overlapped by computation as "exposed". Such exposed time components directly contribute toward execution time. In every design, `conservative` is illustrated on the left, whereas `aggressive` is shown on the right with hatches.

For the `conservative` results, remote memory access time becomes a bottleneck in ❶ due to its low remote memory bandwidth, which stems from the PCIe constraint. When the remote memory bandwidth surpasses 512GB/s, communication is the limiting factor, rather than remote memory access time, in most cases. ❷ outperforms ❸ owing to its superior network bandwidth. ❹ presents the best performance due to the highest remote memory bandwidth and GPU link bandwidth.

With the `aggressive` offloading option, ❶'s training is 2.82× slower, but the performance slowdown decreases to 1.48× in ❹ due to its outstanding remote memory bandwidth. `Aggressive` incurs a minor performance reduction because of high remote memory bandwidth, allowing for a 61.6% per-GPU capacity reduction of the costly HBM component. *This shows that high-bandwidth to remote DDRs opens up an opportunity for aggressive memory offloading, reducing the required local HBM capacity*. Interestingly, `aggressive` enables ❸ to surpass ❷ in performance, contrasting their `conservative` results. This indicates that CXL-based direct memory expansion performs comparably to NVLink-based memory systems when memory bandwidth requirements are low but performs worse when the requirement is high.

VI. Conclusion

In this study, we evaluated four memory system design options in distributed training to tackle the GPU memory wall. Our evaluation revealed several key findings. Firstly, remote memory access time and communication time emerge as major performance bottlenecks when training MoE models. Secondly, the `aggressive` offloading option reduces local HBM memory requirements by 61.6%. Lastly, we observe that per-GPU memory expansion architectures will be able to deliver comparable performance to high-end integrated chip systems when memory bandwidth requirement is low (`conservative`). We plan to continue this study to cover pooled memory systems with more workloads and parallelisms.

REFERENCES

[1] E. Amaro, C. Branner-Augmon, Z. Luo, A. Ousterhout, M. K. Aguilera, A. Panda, S. Ratnasamy, and S. Shenker, "Can Far Memory Improve Job Throughput?" in *Proceedings of the 15th European Conference on Computer Systems (EuroSys)*, 2020.

[2] R. Anil, V. Gupta, T. Koren, and Y. Singer, "Memory-efficient adaptive optimization for large-scale learning," *arXiv preprint arXiv:1901.11150*, vol. 4, 2019.

[3] S. Ö. Arık, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Ng, J. Raiman *et al.*, "Deep Voice: Real-time Neural Text-to-Speech," in *International Conference on Machine Learning*. PMLR, 2017, pp. 195–204.

[4] "Memory Expansion," https://www.asteralabs.com/applications/memory-expansion/, [Online; accessed 4-May-2023].

[5] P. Belevich, Y. Zhao, S. Li, J. Choi, R. Varma, P. Damania, G. Chauhan, M. Yadav, P.-Y. Aquilanti, and S. Ranganatha, "Training a 1 Trillion Parameter Model With PyTorch Fully Sharded Data Parallel on AWS," https://medium.com/pytorch/training-a-1-trillion-parameter-model-with-pytorch-fully-sharded-data-parallel-on-aws-3ac13aa96cff, 2022, [Online; accessed 28-June-2022].

[6] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving," in *Proceedings of the IEEE international conference on computer vision*, 2015.

[7] T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training deep nets with sublinear memory cost," *arXiv preprint arXiv:1604.06174*, 2016.

[8] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv preprint arXiv:1406.1078*, 2014.

[9] S. Choi, T. Kim, J. Jeong, R. Ausavarungnirun, M. Jeon, Y. Kwon, and J. Ahn, "Memory Harvesting in Multi-GPU Systems with Hierarchical Unified Virtual Memory," in *Proceedings of USENIX Annual Technical Conference (ATC)*, 2022.

[10] D. E. Comer and J. Griffioen, "A new design for distributed systems: The remote memory model," 1990.

[11] "NVIDIA DGX-2," https://www.nvidia.com/en-gb/data-center/dgx-2/, [Online; accessed 3-May-2023].

[12] "Intel HPC Roadmap: 800W Rialto Bridge GPU, Falcon Shores XPU, Ponte Vecchio with HBM Benchmarks," https://www.tomshardware.com/news/intel-hpc-roadmap-800w-rialto-bridge-gpu-falcon-shores-xpu-ponte-vecchio-with-hbm, [Online; accessed 3-May-2023].

[13] W. Fedus, B. Zoph, and N. Shazeer, "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity," 2021.

[14] "Habana Gaudi," https://habana.ai/products/, [Online; accessed 3-May-2023].

[15] C. A. Gomez-Uribe and N. Hunt, "The Netflix Recommender System: Algorithms, Business Value, and Innovation," *ACM Transactions on Management Information Systems (TMIS)*, vol. 6, no. 4, pp. 1–19, 2015.

[16] J. Gu, Y. Lee, Y. Zhang, M. Chowdhury, and K. G. Shin, "Efficient Memory Disaggregation with Infiniswap," in *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017.

[17] L. Iftode, K. Li, and K. Petersen, "Memory servers for multicomputers," in *Digest of Papers. Compcon Spring*. IEEE, 1993, pp. 538–547.

[18] 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," *arXiv preprint arXiv:2001.08361*, 2020.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[20] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen, "GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding," *arXiv preprint arXiv:2006.16668*, 2020.

[21] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch, "Disaggregated Memory for expansion and Sharing in Blade Servers," in *Proceedings of the 36th International Symposium on Computer Architecture (ISCA)*, 2009.

[22] V. Nitu, B. Teabe, A. Tchana, C. Isci, and D. Hagimont, "Welcome to Zombieland: Practical and Energy-efficient Memory Disaggregation in a Datacenter," in *Proceedings of the 13th European Conference on Computer Systems (EuroSys)*, 2018.

[23] "NVIDIA Grace Hopper Superchip," https://www.nvidia.com/en-us/data-center/grace-hopper-superchip/, [Online; accessed 14-October-2022].

[24] M. Ott, S. Shleifer, M. Xu, P. Goyal, Q. Duval, and V. Caggiano, "Fully Sharded Data Parallel: faster AI training with fewer GPUs," https://engineering.fb.com/2021/07/15/open-source/fsdp/, 2021, [Online; accessed 28-June-2022].

[25] X. Peng, X. Shi, H. Dai, H. Jin, W. Ma, Q. Xiong, F. Yang, and X. Qian, "Capuchin: Tensor-based GPU Memory Management for Deep Learning," in *Proceedings of the 20th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2020.

[26] S. Rajbhandari, C. Li, Z. Yao, M. Zhang, R. Y. Aminabadi, A. A. Awan, J. Rasley, and Y. He, "DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale," *arXiv preprint arXiv:2201.05596*, 2022.

[27] 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*, 2020.

[28] 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*, 2021.

[29] 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 *USENIX Annual Technical Conference (ATC)*, 2021.

[30] Z. Ruan, M. Schwarzkopf, M. K. Aguilera, and A. Belay, "AIFM: High-Performance, Application-Integrated Far Memory," in *Proceedings of the 14th Symposium on Operating Systems Design and Implementation (OSDI)*, 2020.

[31] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer," *arXiv preprint arXiv:1701.06538*, 2017.

[32] N. Shazeer and M. Stern, "Adafactor: Adaptive learning rates with sublinear memory cost," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4596–4604.

[33] L. Wang, J. Ye, Y. Zhao, W. Wu, A. Li, S. L. Song, Z. Xu, and T. Kraska, "SuperNeurons: Dynamic GPU Memory Management for Training Deep Neural Networks," in *Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, 2018.

[34] S. Williams, A. Waterman, and D. Patterson, "Roofline: an insightful visual performance model for multicore architectures," *Communications of the ACM*, vol. 52, no. 4, pp. 65–76, 2009.

[35] W. Won, T. Heo, S. Rashidi, S. Sridharan, S. Srinivasan, and T. Krishna, "ASTRA-sim2.0: Modeling hierarchical networks and disaggregated systems for large-model training at scale," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2023.

[36] Y. Zhao, R. Varma, C.-C. Huang, S. Li, M. Xu, and A. Desmaison, "Introducing PyTorch Fully Sharded Data Parallel (FSDP) API," https://pytorch.org/blog/introducing-pytorch-fully-sharded-data-parallel-api/, 2022, [Online; accessed 28-June-2022].