CPP: Clustered NPU-PIM Heterogeneous Computing Topology for LLM Inference

2025 International Technical Conference on Circuits/Systems, Computers, and Communications (ITC-CSCC) | 979-8-3315-5363-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/ITC-CSCC66376.2025.11137596

Dongwook Kim
Department of Electrical and Electronic Engineering
Yonsei University
Seoul, South Korea
dwkim3852@yonsei.ac.kr

Dohyun Kim
Department of Electrical and Electronic Engineering
Yonsei University
Seoul, South Korea
middk@yonsei.ac.kr

Eui-Young Chung
Department of Electrical and Electronic Engineering
Yonsei University
Seoul, South Korea
eychung@yonsei.ac.kr

***Abstract*—Transformer inference system can take advantage of heterogeneous computing architectures using Neural Processing Unit (NPU) and Processing-in-Memory (PIM) devices to accelerate GEneral Matrix Multiplication (GEMM) and GEneral Matrix-Vector Multiplication (GEMV) operations each (hereafter NPU-PIM). Several studies on NPU-PIM system showed significant improvement in terms of GEMM and GEMV computation acceleration, but didn't focus on improving communication latency overhead. From our observation based on simulation, NPU-PIM topology is important for communication overhead because it affects the communication dataflow. Dataflow in communication process is related with GEMM/GEMV computation sequence and aggregation of partial results that are distributed for parallel processing. Alternating GEMM/GEMV layers in transformer inference incurs costly data movement between NPUs and PIMs. We found that improvements to the NPU-PIM topology could mitigate the overhead caused by this dataflow. Based on this observation, we propose CPP, an NPU-PIM heterogeneous computing topology for efficient inference of transformer-based models. CPP reduces communication latency overhead by clustering NPUs and PIMs. Clustering leads to separation of dataflow by intra-cluster communication and inter-layer communication. Thereby minimizing resource contention caused by distributed data aggregation and remote memory access. Simulation results show that replacing from the baseline NVIDIA DGX-like crossbar switch topology to CPP reduces communication latency overhead by 35.5%. Also, reduced communication latency leads to the throughput of communication process by 54.9%.**

***Keywords*— Large language model (LLM), Transformer, Processing-in-memory (PIM), Neural processing unit (NPU), Heterogeneous computing**

I. INTRODUCTION

Large language models (LLMs) have become the core infrastructure of the modern AI ecosystem, driving innovation across industries such as chat service, code generation, and other numerous AI services. In particular, models based on transformer architecture have seen continuous performance improvements, as seen in the BERT and GPT series, owing to their long-context processing and parallel reasoning capabilities. The evolution of transformer-based large-scale language models involves exponential growth in model size to improve performance and enable long-context processing [2], and the resulting explosion in compute and memory resource requirements is a key challenge. As a result, there has been a lot of research to accelerate transformer models by leveraging specialized HW that tailored to specific computation operation [15-19].

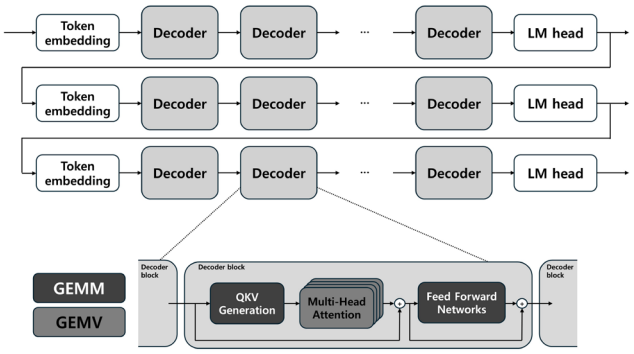

Fig. 1. Inference process of GPT and a decoder block

The operation of GEMM has evolved to be handled by NPUs and high-performance processing units, including the popular GPUs. On the other hand, the other kind of operation that makes up transformer inference, the GEMV operation, is not suitable for these HWs due to its memory-bound nature. Rather, GEMV operation is efficiently accelerated by PIM. Consequently, there have been studies to accelerate both operations using NPU-PIM heterogeneous computing environments [1-3]. In terms of end-to-end inference latency, this NPU-PIM heterogeneous computing environment has been shown to improve throughput by up to 4.84x compared to the GPU-based system, particularly due to the acceleration of the attention layer GEMV operation using PIM [2]. On the other hand, the communication latency overhead caused by layer transitions was independent of the computation time reduction due to the use of NPUs and PIM. Rather, as computation time decreased, communication became a larger part of the overall latency. In an environment using GEMM/GEMV heterogeneous computing acceleration, the communication latency based on the LLAMA 65B model was found to account for about 40% of the total latency [2].

Transformer inference is characterized by alternating and repeating GEMM/GEMV computation. Therefore, data transfer between NPUs and PIMs is required each time the computation type is switched. Meanwhile, multi-head attention (MHA) layer operations can be decomposed into head units and distributed to multiple PIMs for parallel processing. For example, communication latency is observed during the process of combining the results of distributed operations and moving them to the NPU for projection layer, which is a GEMM operation. This communication latency overhead is exacerbated in strategies that use batching to process multiple requests in parallel. Experiments using the

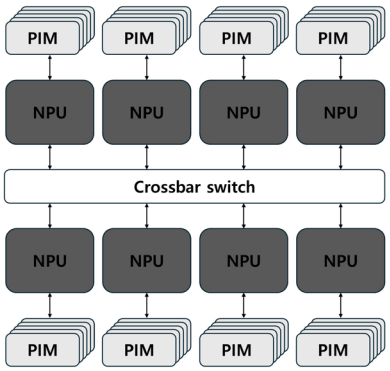

Fig. 2. Baseline DGX-like crossbar switch NPU-PIM topology

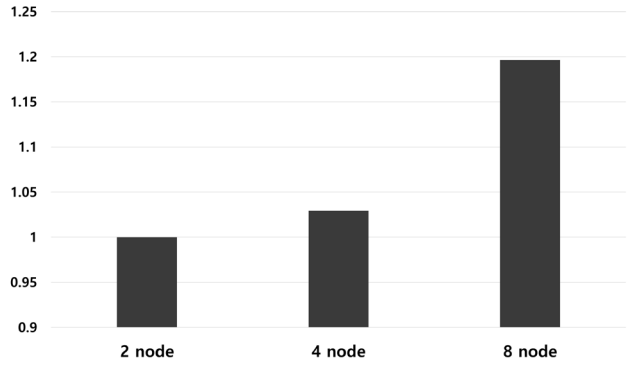

Fig. 3. Average memory access latency of NPU in DGX-like topology with different NPU nodes count

LLAMA 30B model have shown that increasing batch size leads to an overall latency increase of up to 2.6x due to increased competition for compute and memory bandwidth resources between requests [25]. Note that in the existing NPU-PIM studies, performance comparisons are made with NVIDIA DGX systems, and the interconnection topology for NPU and PIM is set to DGX-like crossbar switch [26].

We focused on a solution that would improve communication latency without losing the overall compute time reduction that comes with introducing NPU-PIM heterogeneous computing. Solution that improves communication latency overhead while maintains the overall compute time reduction from the introduction of NPU-PIM was mainly considered. Since the communication dataflow arises from the placement relationship of NPUs and PIMs, the solution is to improve the computing resource topology. Furthermore, to better support distributed parallel computation to efficiently exploit increasingly large models, a topology that minimizes the non-uniform memory access (NUMA) impact of remote memory accesses is needed.

In this regard, we propose CPP, a clustered NPU-PIM computing architecture topology. CPP is the topology for a NPU-PIM environment that employs a batching strategy. This topology has been shown to be efficient in terms of data movement between NPUs and PIMs in the computation transition from GEMM to GEMV and vice versa. The baseline NPU-PIM crossbar switch topology, DGX-like crossbar switch is shown in Fig. 2. This topology is not suitable for reducing communication latency overhead due to the distributed remote PIM channel access. This is because, due to batching and MHA layer operations, the heads of multiple requests are distributed across multiple PIMs, and these heads need to be concatenated. Due to batching and MHA layer operations, the heads of multiple requests are distributed to multiple PIMs, and these heads need to be concatenated. Therefore, the number of remote memory accesses to the NPU increases and multiple heads compete for limited NPU link and crossbar link resources, which ultimately leads to increased communication latency. In experiments using simulation, we observed that in a crossbar switch topology environment, the memory access latency of the NPU increased as the number of NPU nodes connected to the crossbar switch increased. In an environment with 4 PIM devices are located in a single NPU node, the memory access latency increased by 16.2% when there were 8 NPU nodes compared to a total of 4 NPU nodes.

Thus, we propose CPP, a clustered NPU-PIM topology. NPUs and PIMs form single cluster and connected through central I/O die to minimize remote memory access. Layer operations can be isolated within a cluster and this leads to separated intra-cluster and inter-cluster computation. Compared with DGX-like crossbar switch topology, CPP reduces both remote memory access and link contention between NPUs. Therefore, communication latency overhead, which did not improved with DGX-like topology, is reduced significantly. Evaluation using simulation shows that the proposed CPP topology achieves a 35.5% improvement in communication latency overhead compared to the baseline DGX-like crossbar switch topology, leading to a 54.9% improvement in communication throughput.

The extant NPU-PIM heterogeneous computing studies for LLM inference have predominantly focused on the PIM architecture [2, 3] or on data scheduling between the NPU and the PIM [1, 3]. Consequently, this study is the first to concentrate on latency overhead improvement through NPU-PIM computing architecture topology.

II. Background

A. Computational characteristics of transformer-based LLM inference and its accelration

The stack of decoder blocks is a key feature of transformer-based LLM inference in terms of computation [5]. Each decoder block consists of three main layers: query-key-value (QKV) generation, MHA, and feed-forward network (FFN). In a system that employs batching strategy, computation is performed such that the weights for multiple requests are shared, resulting in GEMM operations between the weight and activation matrices. In contrast, the multi-head attention layer requires multiplication between activation matrices and activation vectors where no data are reused, resulting in GEMV operations. Fig. 1 represents overall inference process and internal process of decoder block. Overall inference operation consists of alternating GEMM and GEMV operation step. Moreover, algorithmic structure limits those two computation's parallel execution and they should be done in sequential manner.

NPUs can be designed and utilized to accelerate GEMM which has computation bounded features [20, 21], and it can be applied to transformer-based models inference for GEMM computation layers like QKV generation and FFN. While NPU is designed for compute-intensive tasks, accelerating GEMV operations using NPU is not appropriate in terms of HW utilization because of memory-bound features of GEMV. For GEMV computation acceleration, PIM technology can be applied, as it is suitable for memory-bound and bandwidth-intensive tasks [10, 22, 23]. PIM embeds the logic unit inside DRAM so that computation can be performed inside the memory device without processing unit's memory access for fetching data, thus reducing

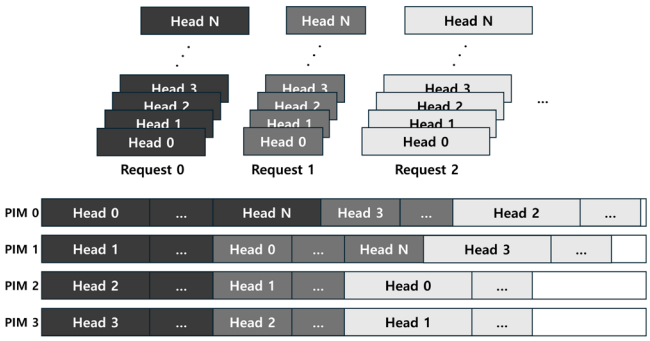

Fig. 4. Mapping of GEMV computation on PIM

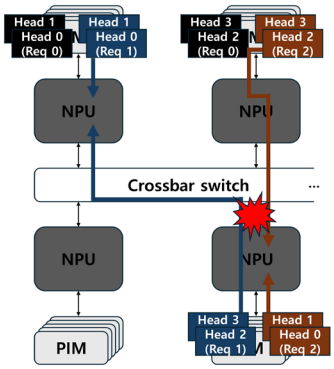

Fig. 5. Contention caused by aggregation of distributed data

expensive DRAM access [24]. Conversely, using PIM for computer-intensive GEMM operation is not efficient.

B. Transformer-based model inference

A key component of the Transformer model is MHA [5], a mechanism for processing input data in parallel in multiple representational spaces, where each attention head can extract different semantic relationships or patterns in the input sequence. The nature of multi-head attention is such that each head can be considered an independent computational unit, allowing for "head-wise" parallelism. Each head produces a different projection of the input data and computes its own attentional patterns independently. This head-specific computation can be distributed across multiple computational units and processed in parallel.

In real-world service environments with large language models, batching processing techniques are widely used to process not only single requests but also multiple user requests simultaneously. Batch processing bundles multiple input sequences into a single computation group to fully utilize the parallel processing power of the computing device, thereby improving the overall system throughput. However, this batch processing introduces contention for memory resources, especially when requests with different sequence lengths share the same memory device, each request competes for the memory bandwidth and capacity it requires.

When parallel processing of these multi-headed attentions is combined with batch-based multi-request processing, an aggregation process is essential to consolidate the results of the distributed computations. The outputs of each attention layer must be concatenated for the next layer. This aggregation process incurs communication overhead of collecting and recombining data distributed across multiple computational units. Since the amount of data to be aggregated increases as the batch size increases, and the communication complexity on the network topology

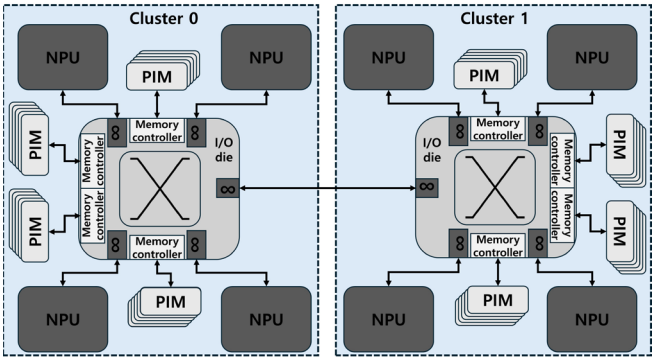

Fig. 6. Proposed clustered NPU-PIM computing topology, CPP

increases as the number of heads increases, efficient management of this aggregation process emerges as an important challenge for distributed inference systems in transformer-based models.

III. Limitation of Crossbar Switch NPU-PIM

In this section, we discuss the baseline NPU-PIM topology, the DGX-like crossbar switch. We show that this topology introduces a communication latency overhead in transformer-based model inference.

The NVIDIA DGX system represents a high-performance computing platform specifically engineered for AI and deep learning workloads. Each of the 8 GPUs has 5 HBM stacks as local memory. Thus, the system utilizes a total of 40 HBM devices, with a capacity of 320GB or 640GB. Each GPU is connected to all other GPUs via a crossbar switch called NVSwitch. As assumed in the previous NPU-PIM heterogeneous computing architecture, the baseline interconnection topology was set to a DGX-like crossbar switch as shown in Fig. 2.

The advantage of the crossbar switch topology is that all NPUs can be connected to each other. In situations where an NPU needs data from the local memory of another NPU, the crossbar switch can reduce the latency overhead of remote memory access by reducing the hop count required. Also, the process of passing the results of computation on one NPU to another NPU can be done with less overhead by the same reason. However, in LLM inference, where batching is applied to process many requests simultaneously, distributed PIM and crossbar switches become the bottleneck due to competition for limited memory bandwidth and capacity. In simulations using gem5 [13], it was observed that the average memory access latency of NPUs in a DGX-like crossbar switch topology environment degrades as the number of NPU nodes connected to the crossbar switch increases. This result can be seen in Fig. 3, which shows that the average memory access latency of each NPU increased by 16.2% when there are 8 NPU nodes compared to 4 NPU nodes. In the experiment the latency of the crossbar switch is assumed to be constant even when the number of connected nodes changes. Thus this result explains the contention caused by different requests on the NPU link due to increased remote memory access.

Distributed parallelization of heads due to multi-head attention exacerbates this problem. A greedy or round-robin approach to mapping heads to PIMs is used [2, 3], which is an appropriate strategy to maximize the utilization of PIMs and minimize the GEMV load imbalance between PIMs. Consequently, PIM device is allowed to have a mix of heads

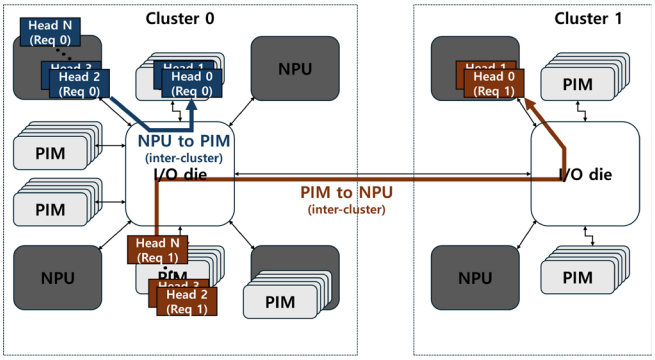

Fig. 7. Communication dataflow on CPP topology

belonging to different requests, as shown in Fig. 4. The problem is that this mapping causes the heads to be more distributed than if they were mapped to specific PIMs. The distribution of heads increases communication overhead and memory resource contention because it increases remote PIM access during concatenation after attention layer operations, as shown in Fig. 5. From the evaluation results of [2], accelerating attention layer computation using PIM is very effective, with up to 3.91x improvement in end-to-end inference latency on GPT-3 175B model. Also, in DGX-like topologies, there is a latency penalty due to the NUMA structure, which makes it difficult to apply distributed parallel computation in increasingly large models. Therefore, it is necessary to reduce the communication overhead while maintaining the round-robin head mapping method for high PIM utilization.

IV. CPP: Clustered NPU-PIM

To address the communication overhead problem due to distributed multi-head attention processing using PIM while maintaining the high utilization of PIM with round-robin head mapping, we propose CPP, a clustered NPU-PIM topology. The overall structure of this topology is shown in Fig. 6.

Contention caused by communication process and remote PIM access can be resolved by grouping the NPUs and PIMs. In our proposed CPP topology, 4 NPUs and their accompanying PIMs form a single cluster. At the center of the cluster is an IO die, which mediates memory accesses and NPU-to-PIM communication. The IO die also mediates connectivity between clusters, allowing other clusters to access memory and move data leading to separated intra-cluster communication and inter-cluster communication. This communication dataflow mechanism is shown in Fig. 7. By isolating the successive layer operations that require switching between GEMM and GEMV to communicate within a cluster, it is ensured that communication only takes place within one cluster. In addition, in the DGX-like topology, remote memory access must go through the crossbar switch and NPU link, causing additional contention. However in CPP, the path of data reading and writing from PIM within a cluster can be separated from the NPU link. Therefore, unnecessary link contention is minimized. It is also easy to increase scalability using CPP topology as it is done by simply adding more clusters. Increasing scalability by adding cluster minimizes performance degradation that can be caused by bigger communication scale, as operation can be isolated within cluster. Also, latency penalty due to NUMA is minimized as PIM can reduce costly memory access.

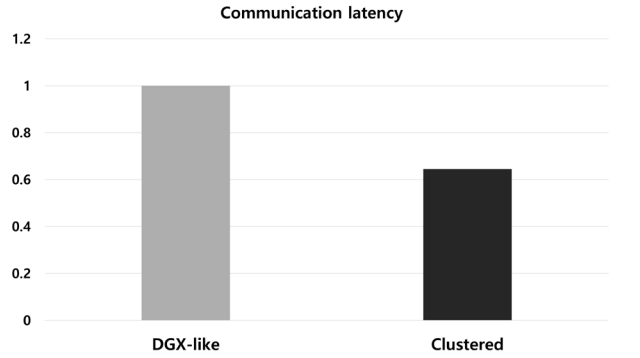

Fig. 8. Normalized communication latency of DGX-like topology and clustered topology

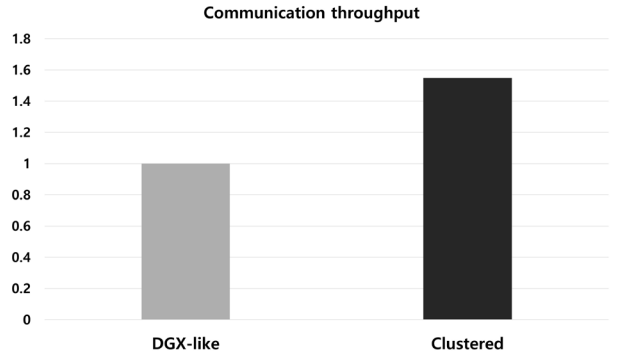

Fig. 9. Normalized communication throughput of DGX-like topology and clustered topology

V. Methodology

A. Experimental environment

To measure the performance improvement in the communication step due to the change of NPU-PIM topology, we used in-house simulators based on Scale-sim [11, 12] and gem5. We utilized Scale-sim to generate memory access traces of workload and used them to compare the performance in terms of data movement across NPU-PIM topologies.

The NPU-PIM system architecture was modeled using gem5. The proposed CPP topology is intended to be compared to the crossbar switch topology of NVIDIA DGX. The system configuration is similar to the DGX A100, which is configured with 8 GPUs, and 5 HBM stack per GPU, i.e., 40 HBMs in total. Our NPU-PIM system consists of 8 NPUs and 4 PIMs for each NPU, totaling 32 PIMs. The difference in the number of memory nodes reflects a modification due to simulation constraints in gem5 and does not affect the fundamental architectural topology and communication patterns.

B. Workload

We utilize transformer-based LLM, GPT-2 [14], and focused on the transitioning from GEMV computation layer to GEMM computation layer. This is the process of switching from PIM operation to NPU operation, and it is suitable to check the communication overhead because the heads that are distributed across multiple PIM devices need to be concatenated for projection layer operation on NPU.

VI. Evaluation

Simulation using gem5 were conducted to evaluate the performance of the proposed CPP topology. CPP topology is

compared with DGX-like crossbar switch topology. Fig. 8 shows the results of the normalized latency overhead for both topologies. CPP topology has less communication latency overhead than the DGX-link crossbar switch topology. Specifically, The latency overhead in the CPP topology was found to be 64.5% of that in the DGX-like topology, resulting in a 35.5% communication latency overhead improvement. This improvement in latency overhead also led to an increase in communication throughput, which increased by 54.9% when NPU-PIM topology was CPP, compared to the DGX-like topology scenario. This result is shown in Fig. 9. It can thus be concluded that the selection of an appropriate NPU-PIM topology has the potential to reduce the communication latency between the NPU and the PIM due to the switching between GEMM and GEMV operations.

VII. Conclusion

We found that previous attempts to accelerate transformer-based LLM inference using a combination of NPUs and PIMs were meaningful in terms of GEMM and GAMV computation acceleration. However communication latency overhead did not improved with no consideration for the topology architecture of heterogeneous computing platform. Data need to move between the NPU and PIM multiple times depending on computation type. Moreover, in batched inference system where multiple requests are processed simultaneously, communication process poses bigger issue. Our observation is that DGX-like crossbar switch NPU-PIM topology suffers from memory resource contention between multiple requests. Thus it is not suitable for communication process between GEMM and GEMV layers. Based on this observations, we propose CPP, a clustered topology architecture for NPU-PIM. CPP topology organizes a cluster of NPUs and PIMs to reduces competition for limited memory bandwidth and link resources. This is because CPP can separate and isolate the communication dataflow into intra-cluster and inter-cluster communication. Experiment with simulation using Scale-sim and gem5 is done to compare the baseline DGX-like topology and CPP topology. The results show that CPP achieves a latency overhead improvement of 35.5% compared with DGX-like one. Additionally communication throughput is also improved by 54.9%, validating our observation that computing architecture topology matters in terms of communication overhead in LLM inference using NPU-PIM.

Acknowledgment

This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2022-0-00050, Development of PIM Computing Architecture based on Data-Flow) and the Technology Innovation Program (No. RS-2024-004020541, Development of CXL-based PNM Architecture and Simulation Platform for LLM Acceleration) funded By the Ministry of Trade, Industry and Energy (MOTIE, Korea).

References

[1] M. Seo et al., “IANUS: Integrated Accelerator based on NPU-PIM Unified Memory System,” Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3, pp. 545–560, Apr. 2024, doi: https://doi.org/10.1145/3620666.3651324

[2] J. Park et al., “AttAcc! Unleashing the Power of PIM for Batched Transformer-based Generative Model Inference,” Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, pp. 103–119, Apr. 2024, doi: https://doi.org/10.1145/3620665.3640422

[3] G. Heo et al., “NeuPIMs: NPU-PIM Heterogeneous Acceleration for Batched LLM Inferencing,” Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3, pp. 722–737, Apr. 2024, doi: https://doi.org/10.1145/3620666.3651380

[4] J. Cho, M. Kim, H. Choi, G. Heo and J. Park, "LLMServingSim: A HW/SW Co-Simulation Infrastructure for LLM Inference Serving at Scale," 2024 IEEE International Symposium on Workload Characterization (IISWC), Vancouver, BC, Canada, 2024, pp. 15-29, doi: 10.1109/IISWC63097.2024.00012

[5] A. Vaswani et al., “Attention is All you Need,” in NeurIPS, 2017. doi: https://dl.acm.org/doi/10.5555/3295222.3295349

[6] M. Zhou, W. Xu, J. Kang and T. Rosing, "TransPIM: A Memory-based Acceleration via Software-Hardware Co-Design for Transformer," 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Seoul, Korea, Republic of, 2022, pp. 1071-1085, doi: 10.1109/HPCA53966.2022.00082

[7] C. Li, Z. Zhou, S. Zheng, J. Zhang, Y. Liang, and G. Sun, “SpecPIM: Accelerating Speculative Inference on PIM-Enabled System via Architecture-Dataflow Co-Exploration,” Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3, pp. 950–965, Apr. 2024, doi: https://doi.org/10.1145/3620666.3651352

[8] S. Yun et al., "Duplex: A Device for Large Language Models with Mixture of Experts, Grouped Query Attention, and Continuous Batching," 2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO), Austin, TX, USA, 2024, pp. 1429-1443, doi: 10.1109/MICRO61859.2024.00105

[9] H. Kwon et al., “LoL-PIM: Long-Context LLM Decoding with Scalable DRAM-PIM System,” 2024, arXiv preprint, dio: https://doi.org/10.48550/arXiv.2412.20166

[10] M. He et al., "Newton: A DRAM-maker’s Accelerator-in-Memory (AiM) Architecture for Machine Learning," 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Athens, Greece, 2020, pp. 372-385, doi: 10.1109/MICRO50266.2020.00040

[11] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "Scale-sim: Systolic cnn accelerator simulator," 2018, arXiv preprint doi: https://doi.org/10.48550/arXiv.1811.02883

[12] A. Samajdar, J. M. Joseph, Y. Zhu, P. Whatmough, M. Mattina and T. Krishna, "A Systematic Methodology for Characterizing Scalability of DNN Accelerators using SCALE-Sim," 2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Boston, MA, USA, 2020, pp. 58-68, doi: 10.1109/ISPASS48437.2020.00016

[13] N. Binkert et al., “The gem5 simulator,” SIGARCH Comput. Archit. News 39, 2 (May 2011), 1–7. doi: https://doi.org/10.1145/2024716.2024718

[14] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” OpenAI blog, 1(8):9, 2019

[15] S. Hong et al., “DFX: A Low-Latency Multi-FPGA Appliance for Accelerating Transformer-Based Text Generation,” Proceedings of the 55th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2022, pp. 616–630, doi: https://doi.org/10.1109/MICRO56248.2022.00051

[16] H. Wang, Z. Zhang and S. Han, "SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning," 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Seoul, Korea (South), 2021, pp. 97-110, doi: 10.1109/HPCA51647.2021.00018

[17] R. Kaplan, "Intel Gaudi 3 AI Accelerator: Architected for Gen AI Training and Inference," 2024 IEEE Hot Chips 36 Symposium (HCS), Stanford, CA, USA, 2024, pp. 1-16, doi: 10.1109/HCS61935.2024.10665178

[18] T. Jeong and E. -Y. Chung, "PipePIM: Maximizing Computing Unit Utilization in ML-Oriented Digital PIM by Pipelining and Dual Buffering," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 43, no. 12, pp. 4585-4598, Dec. 2024, doi: 10.1109/TCAD.2024.3410842

[19] C. Guo et al,. “OliVe: Accelerating Large Language Models via Hardware-friendly Outlier-Victim Pair Quantization,” Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA), New York, NY, USA, 2023, Article 3, pp. 1–15. https://doi.org/10.1145/3579371.3589038

[20] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. 2016. Eyeriss: a spatial architecture for energy-efficient dataflow for convolutional neural networks. SIGARCH Comput. Archit. News 44, 3 (June 2016), 367–379. https://doi.org/10.1145/3007787.3001177

[21] H. Sharma et al., "Bit Fusion: Bit-Level Dynamically Composable Architecture for Accelerating Deep Neural Network," 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), Los Angeles, CA, USA, 2018, pp. 764-775, doi: 10.1109/ISCA.2018.00069

[22] F. A. Siddique et al., "Architectural Modeling and Benchmarking for Digital DRAM PIM," 2024 IEEE International Symposium on Workload Characterization (IISWC), Vancouver, BC, Canada, 2024, pp. 247-261, doi: 10.1109/IISWC63097.2024.00030

[23] S. Lee et al., "Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology : Industrial Product," 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), Valencia, Spain, 2021, pp. 43-56, doi: 10.1109/ISCA52012.2021.00013

[24] D. Patterson et al., "A case for intelligent RAM," in IEEE Micro, vol. 17, no. 2, pp. 34-44, March-April 1997, doi: 10.1109/40.592312 D. Patterson et al., "A case for intelligent RAM," in IEEE Micro, vol. 17, no. 2, pp. 34-44, March-April 1997, doi: 10.1109/40.592312

[25] B. Sun et al, “Llumnix: dynamic scheduling for large language model serving,” Proceedings of the 18th USENIX Conference on Operating Systems Design and Implementation, USA, 2024, Article 10, 173–191

[26] NVIDIA, "NVIDIA DGX A100." Data-Center/nvidia-dgx-a100-datasheet.pdf (2022)