Pyramid: Accelerating LLM Inference with Cross-Level Processing-in-Memory

Liang Yan, Xiaoyang Lu, Xiaoming Chen, Yinhe Han, and Xian-He Sun, *Fellow, IEEE*

***Abstract*—Integrating processing-in-memory (PIM) with GPUs accelerates large language model (LLM) inference, but existing GPU-PIM systems encounter several challenges. While GPUs excel in large general matrix-matrix multiplications (GEMM), they struggle with small-scale operations better suited for PIM, which currently cannot handle them independently. Additionally, the computational demands of activation operations exceed the capabilities of current PIM technologies, leading to excessive data movement between the GPU and memory. PIM's potential for general matrix-vector multiplications (GEMV) is also limited by insufficient support for fine-grained parallelism. To address these issues, we propose Pyramid, a novel GPU-PIM system that optimizes PIM for LLM inference by strategically allocating cross-level computational resources within PIM to meet diverse needs and leveraging the strengths of both technologies. Evaluation results demonstrate that Pyramid outperforms existing systems like NeuPIM, AiM, and AttAcc by factors of 2.31×, 1.91×, and 1.72×, respectively.**

***Index Terms*—Large language models, Processing-in-memory**

I. LLM Inference in GPU-PIM Systems

LARGE language models (LLMs) have emerged as powerful machine learning tools capable of solving many natural language processing tasks. As shown in Fig. 1, the inference process of LLMs can be divided into two conceptual stages: summarization and generation. In the summarization stage, token embedding converts input tokens into dense vectors, which are then refined by transformer decoder blocks to process the embeddings and capture long-range dependencies. This phase ends with generating the first token. In the generation stage, the transformer decoder blocks continue processing embeddings of previously generated tokens and transform these embeddings into subsequent output tokens. The entire inference process relies heavily on the transformer decoder blocks, which are critical for capturing context and relationships across the sequence. Each transformer decoder block consists of three primary layers: (1) query-key-value (QKV) generation, (2) multi-head attention (MHA), and (3) a set of feed-forward networks (FFNs). These layers involve intensive matrix operations, including general matrix multiplication (GEMM) and matrix-vector multiplication (GEMV).

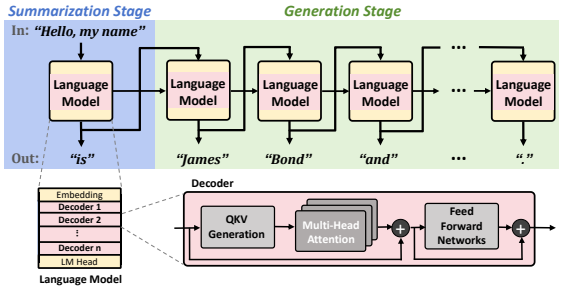

Fig. 1. Illustration of LLM inference.

Large-scale GEMM operations are ideal for GPUs due to their parallel compute capabilities. In contrast, GEMV operations have lower arithmetic intensity, resulting in underutilized GPU resources [1]. PIM technology, which is effective for memory-bound tasks [2], offers promise for GEMV operations. Current GPU-PIM systems for LLM inference optimize GEMV in the MHA layer during generation [3], [4]. However, several challenges remain that hinder the full potential of PIM in accelerating LLM inference.

High Time to First Token (TTFT). Existing GPU-PIM studies [1] offload all GEMM operations to GPU. However, for short requests, GPUs need to batch multiple requests to ensure efficient utilization. The batching process introduces a delay as short requests must wait until there are enough to form a predefined input matrix before being processed. This delay challenges the reduction of TTFT [5] which is crucial because it directly impacts user experience.

Heavy Data Movement. LLMs also rely heavily on element-wise activation operations, such as the ubiquitous Softmax and GeLU, which are compute-intensive and significantly contribute to the overall computational load of LLM inference. Existing PIM architectures limit their compute capabilities to near-bank processing units, forcing these activation operations to be offloaded to GPU. This offloading incurs significant data movement overheads

Coarse Parallelism. Most existing works [1], [3], [4] focus on offloading GEMV operations in PIM but do not thoroughly explore the parallelism potential that PIM can offer. These studies typically concentrate on near-bank computational units, achieving parallelism primarily at the bank level. However, each bank contains multiple subarrays, and optimizing parallelism at this finer granularity remains underexplored.

To address these challenges, we propose Pyramid, a GPU-

This work was supported in part by National Natural Science Foundation of China under Grant 62488101 and Grant 62495104, and in part by Youth Innovation Promotion Association CAS.

Liang Yan, Xiaoming Chen and Yinhe Han are with Intelligent Computing Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100190, China.

Xiaoyang Lu and Xian-He Sun are with Department of Compute Science, Illinois Institute of Technology, Chicago, IL, USA.

(Corresponding authors: Xiaoming Chen, E-mail: chenxiaoming@ict.ac.cn)

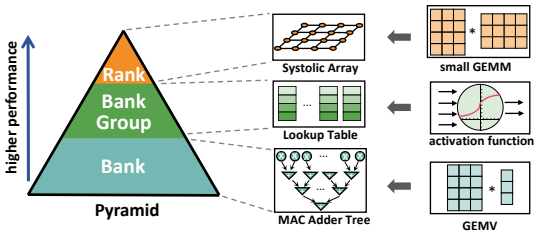

Fig. 2. Illustration of Pyramid computing resource configuration and corresponding operations in LLM inference.

PIM system for LLM inference with an intelligent cross-level PIM architecture . It integrates compute resources across memory levels in PIM, boosting PIM-side power with minimal hardware overhead, rather than relying heavily on GPUs.

II. Pyramid Architecture

A. Overview

Pyramid is an heterogeneous GPU-PIM systems to accelerate LLM inference across a spectrum of operations. As shown in Fig. 2, Pyramid fully leverages three levels of computing resources within PIM. First, small systolic arrays are deployed in ranks to support GEMM operations with short requests. Second, lookup table (LUT)-based modules provide fast activation calculations at the bank group level. Third, parallel MAC units in banks, optimized with fine-grained subarray parallelism, further accelerate GEMVs.

As illustrated in Fig. 3, Pyramid comprises 8 DRAM channels. Each channel contains 2 ranks, with 4 bank groups per rank, 4 memory banks per bank group, and 32 subarrays per bank. This hierarchical organization allows Pyramid to optimize processing at multiple levels, ensuring that each type of operation in LLM inference is handled by the most appropriate hardware resource. Pyramid introduces three distinct types of processing units: RankPIM, BGPIM, and BankPIM, each corresponding to different levels of the memory hierarchy within PIM (rank, bank group, and bank, respectively).

RankPIM is equipped with 4 small 16×16 systolic arrays, specifically designed to accelerate small-scale GEMM operations frequently encountered in LLM inference. By deploying RankPIM, Pyramid can directly execute these operations belong to short request inference in parallel at PIM side, eliminating the need for batching that is typically required for GPU processing, which is critical for reducing TTFT.

BGPIM incorporates LUT-based modules that efficiently handle activation functions in bank group level. These modules incorporate with an interpolation module, and an arithmetic module, as illustrated in Fig. 3. By deploying BGPIM, these operations can be executed on the PIM side, significantly reducing data movement between GPU and memory.

BankPIM features a novel configuration of multiplication accumulator (MAC) units that exploit subarray-level parallelism. This configuration is effective for accelerating GEMV operations, which are prevalent in the attention mechanisms of LLMs and benefit significantly from the fine-grained parallelism provided by PIM.

Pyramid strategically increases PIM computational power from the innermost levels to the outermost levels. It effectively handles small-scale GEMM operations, activation functions, and GEMV operations, all within the PIM subsystem. To seamlessly integrate the system with the existing architecture, we introduce several CUDA-like instructions in the ISA to facilitate task offloading between PIM and GPU.

B. Workflow of Decoder Blocks

Unlike traditional GPU-based or existing PIM-GPU systems, where short requests are delayed to accumulate into batches for efficient GPU processing, Pyramid introduces a novel dataflow for executing decoder blocks that accommodates requests of varying lengths. During both single-batch and multi-batch summarization stages, the system first performs a GEMM operation scale assessment before the QKV generation. This step determines whether the upcoming GEMM operations should be processed by the PIM or offloaded to the GPU, based on a predefined threshold indicated by the roofline model. The threshold for GEMM scale is determined by the request length (number of tokens) and the roofline model, which together establish the GEMM execution threshold before runtime. For instance, in GPT-3 175B, when the request exceeds 192 tokens, the GEMM operation becomes compute-bound, making GPU execution more suitable. Otherwise, the GEMM operation becomes memory-bound, making PIM execution more efficient. If the GEMM operations in QKV generation are better suited for the GPU, all subsequent operations, including those in QKV generation, MHA, and FFNs, are offloaded to the GPU. Otherwise, the small-scale GEMM operations involved in QKV generation and MHA are offloaded to the PIM. In addition, a second GEMM operation scale assessment is then performed before the FFN layer to decide whether the subsequent GEMM operations should continue on the PIM or be offloaded to the GPU.

For the generation stage, the workflows differ between single-batch and batched requests. In the case of a single batch, since only one new token is generated at a time, the matrix operations involved are GEMV operations, all offloaded to PIM. For batched requests, although each individual request generates only one token at a time, multiple tokens can be generated across the batch. This allows for the reuse of weight matrices, enabling GEMM operations in the QKV generation layer and FFN layer. Before executing these steps, it is necessary to assess whether the GEMM operations should be processed by the GPU or PIM. However, in the MHA layer of the multi-batch scenario, where no matrices can be reused, the operations remain GEMV-based and are most efficiently executed on the PIM.

This tailored workflow management in Pyramid, which distinguishes it from traditional GPU-PIM systems for LLM inference, enables the system to respond more promptly to both short and long requests, thereby enhancing its ability to deliver responsive and high-performance LLM inference capabilities.

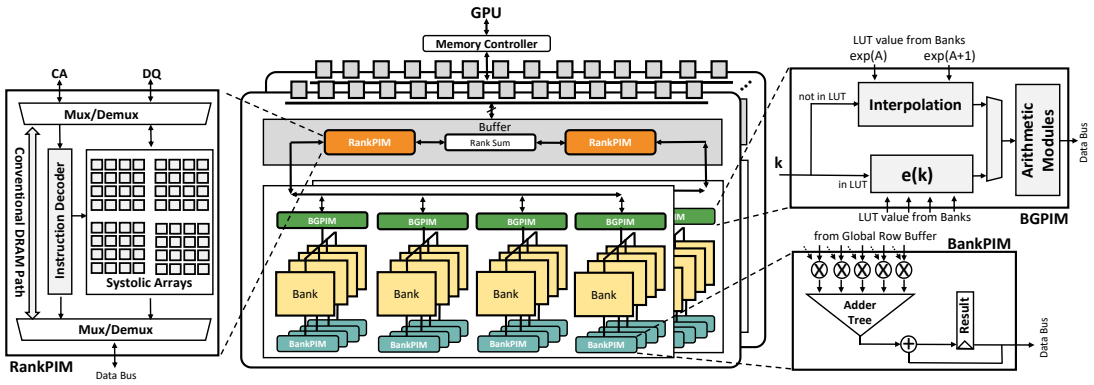

Fig. 3. Organization of the PIM Subsystem in Pyramid Architecture.

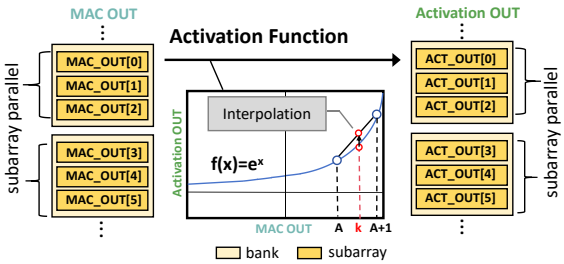

Fig. 4. Activation functions utilizing exponential functions are computed using LUTs, with linear interpolation to increase accuracy.

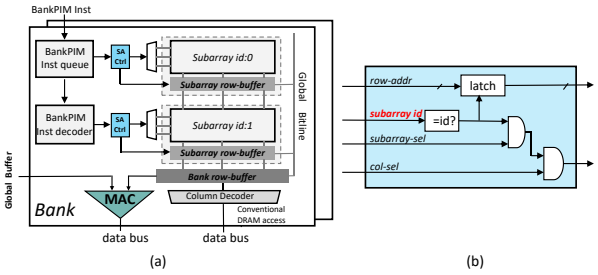

Fig. 5. Pyramid BankPIM with subarray parallelism design. (a) Bank scheme. (b) Subarray access controller.

C. RankPIM: Rank-Level Systolic Arrays

To enhance LLM inference performance within the GPU-PIM system, particularly for short requests, we propose the integration of RankPIM into the rank buffer. As illustrated in Fig. 3, the primary enhancement involves the incorporation of four 16×16 systolic arrays, specifically designed to execute small-scale GEMM operations in parallel. This hardware extension leverages the high internal bandwidth of PIM, offering a significant boost in processing efficiency.

D. BGPIM: Bank Group-Level LUT-Based Modules

Activation operations can be broadly categorized into those that utilize exponential functions (e.g., Softmax, GeLU, Tanh, and Sigmoid) and those that do not (e.g., ReLU and Leaky ReLU). We classify activation functions that use exponential functions as complex activation functions. Due to the limited computational power of current PIM architectures, executing these complex activation functions typically requires offloading to GPUs, resulting in significant data movement between the GPU and memory. Pyramid addresses this limitation by enabling the execution of complex activation functions directly within PIM through BGPIM, utilizing LUT-based modules. To balance memory overhead and accuracy, we use interpolation for values not present in the lookup table and store only exponential function values. Fig. 4 illustrates this process.

E. BankPIM: Bank-Level MACs with Subarray Parallelism

As shown in Fig. 5(a), we place the MAC unit beside the bank row buffer. To achieve subarray-level parallelism, modifications are made to the subarray controller circuit to manage the transition of data from subarry row buffers to bank row buffer, as shown in Fig. 5(b). A specific row is connected to the subarray row buffer upon activation by the row address and subarray ID, and it connects to the bank row buffer only when designated by the subarray selection signal. This design enables simultaneous activation of multiple subarrays within a bank, with temporary storage in subarray row buffers. Building upon this, Pyramid effectively reduces the number of DRAM row activations required for GEMV operations.

III. Evaluation

A. Methodology

We develope an in-house simulator by modifying Ramulator [6] to evaluate the performance of both GPU systems and GPU-PIM systems. Ramulator is a DRAM simulator, has been cross-validated with real DRAM devices and is extensively used in prior research for PIM evaluations [7]. The detailed simulation configuration of Pyramid is presented in Table I.

B. Overall Performance

We compare Pyramid with state-of-the-art GPU-PIM systems for LLM inference, including NeuPIM [1], AiM [3], and AttAcc [4], as shown in Fig. 6. The baseline for comparison was set as GPU+PIM, a PIM-enabled GPU baseline where only the GEMV operations in MHA layers are mapped to PIM, while all other computations are performed on the GPU. The execution times of the QKV, MHA, and FFN layers are

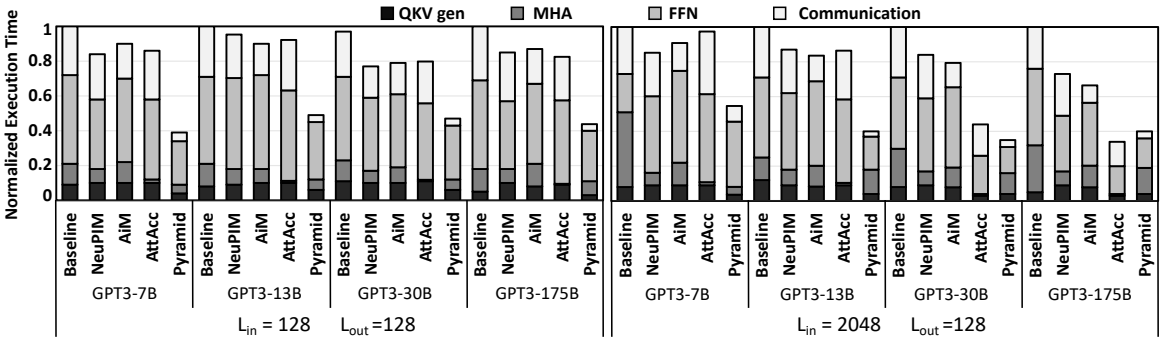

Fig. 6. Normalized decoder execution time of GPT3-7B, GPT3-13B,GPT3-30B and GPT3-175B for various L_{in} and L_{out} to process 1000 request. All results normalized to baseline.

TABLE I
SIMULATED PIM CONFIGURATIONS.

GPU	Nvidia A100-PCIe 80G
PIM System	GDDR6: Channel=8; Ranks/Channel=2; BGs/Rank=4; Banks/BG=4; Subarrays/Bank=32; Capacity/Channel=4GB; Frenquency=1GHz;
Timing Constraint	tRCD=12ns, tRP=12ns, tCCD=1ns, tWR=12ns,tRFC=445ns
Energy and Latency Parameter	DRAM ACT energy=2 nJ, DRAM RD/WR=6.48 pJ/bit, BF16 adder=0.9pJ/Op, BF16 mult=2.4pJ/Op
PIM Processing Units	RankPIM: 4 16x16 systolic arrays per Rank, 1GHz
	BGPIM: Interpolation and Arithmetic Modules per BG
	BankPIM: MAC per Bank, 1GHz

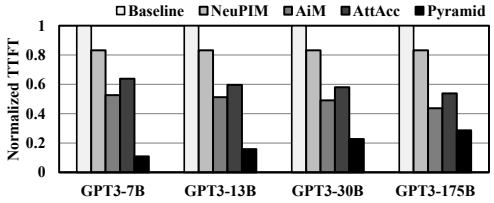

Fig. 7. Normalized TTFT comparison across different model sizes, evaluated with 1000 short requests (L_{in}:128, L_{out}:1) and 1000 long requests (L_{in}:2048, L_{out}:1).

measured using timestamps in our simulator. Communication overhead is evaluated via GPU-PIM data transfers, using a PIM response model to calculate delays for accurate modeling.

We used L_{in} and L_{out} to represent the input and output lengths of a request, respectively. We evaluated the LLM inference decoder execution time across these five GPU+PIM systems with L_{in} set to 128 and 2048. For short requests with L_{in}=128, Pyramid outperformed all four other GPU+PIM systems. Specifically, Pyramid reduced execution time by 55.3%, 47.6%, 48.3%, and 47.5% on average compared to baseline, NeuPIM, AiM, and AttAcc, respectively. These performance improvements are primarily due to Pyramid's capability to handle end-to-end inference on the PIM side for short requests, reducing communication overhead and data movement, particularly for LUT-based activation functions.

For long requests with L_{in}=2048, Pyramid continued to demonstrate significant performance gains, reducing execution time by 57.6%, 48.5%, 47.1%, and 35.2% on average compared to baseline, NeuPIM, AiM, and AttAcc, respectively. Pyramid's workflow is designed to accommodate requests of varying lengths, allowing longer requests to be efficiently accelerated by the GPU.

C. TTFT Optimization

We evaluate the average TTFT across different model sizes to validate the effectiveness of our PIM implementation, specifically its support for small-scale GEMM computations. The average TTFT results are presented in Figure 7, with all values normalized to the baseline. The results demonstrate that Pyramid achieves significantly shorter average TTFT for requests of varying lengths, especially for smaller model sizes.

D. Power and Area Overhead

Pyramid contributes an additional overhead of 12.7% of a DRAM chip under 32nm process, which typically measures around 100 mm^2 [7]. The area overhead of RankPIM is 3.07 mm^2. And BGPIM consumes 0.012 mm^2. BankPIM with 9.62 mm^2 overhead. Furthermore, the power overhead of Pyramid is 47.6 mW, which is considerably lower compared with traditional GPU-PIM system.

IV. CONCLUSION

In this paper, we introduce Pyramid, an innovative GPU-PIM system designed to optimize LLM inference. Pyramid integrates cross-level processing units within PIM. The system is carefully tailored to reduce TTFT, minimize data movement, and enhance parallelism. Our comprehensive evaluations demonstrate that Pyramid consistently outperforms existing solutions, highlighting its significant potential in advancing efficient LLM inference workloads.

REFERENCES

[1] G.Heo et al. Neupims: Npu-pim heterogeneous acceleration for batched llm inferencing. In *ACM ASPLOS*, 2024.

[2] L.Yan et al. Copim: a concurrency-aware pim workload offloading architecture for graph applications. In *IEEE ISLPED*, pages 1–6, 2021.

[3] D.Kwon et al. 1.25 v 8gb 16gb/s/pin gddr6-based accelerator-in-memory supporting 1tflops mac operation and various activation functions for deep learning application. *IEEE JSSC*, 2022.

[4] J.Park et al. Attacc! unleashing the power of pim for batched transformer-based generative model inference. In *ACM ASPLOS*, 2024.

[5] S.Hong et al. Dfx: A low-latency multi-fpga appliance for accelerating transformer-based text generation. In *IEEE MICRO*, 2022.

[6] Y.Kim et al. Ramulator: A fast and extensible dram simulator. *IEEE CAL*, 15(1):45–49, 2015.

[7] G.Dai et al. Dimmining: pruning-efficient and parallel graph mining on near-memory-computing. In *IEEE ISCA*, 2022.