

# Unleashing the Potential of PIM: Accelerating Large Batched Inference of Transformer-Based Generative Models

Jaewan Choi<sup>ID</sup>, Jaehyun Park<sup>ID</sup>, Kwanhee Kyung<sup>ID</sup>, Nam Sung Kim<sup>ID</sup>, *Fellow, IEEE*,  
and Jung Ho Ahn<sup>ID</sup>, *Senior Member, IEEE*

**Abstract**—Transformer-based generative models, such as GPT, summarize an input sequence by generating key/value (KV) matrices through attention and generate the corresponding output sequence by utilizing these matrices once per token of the sequence. Both input and output sequences tend to get longer, which improves the understanding of contexts and conversation quality. These models are also typically batched for inference to improve the serving throughput. All these trends enable the models' weights to be reused effectively, increasing the relative importance of sequence generation, especially in processing KV matrices through attention. We identify that the conventional computing platforms (e.g., GPUs) are not efficient at handling this attention part for inference because **each request generates different KV matrices, it has a low operation per byte ratio regardless of the batch size, and the aggregate size of the KV matrices can even surpass that of the entire model weights.** This motivates us to propose AttAcc, which exploits the fact that the KV matrices are written once during summarization but used many times (proportional to the output sequence length), each multiplied by the embedding vector corresponding to an output token. The volume of data entering/leaving AttAcc could be more than orders of magnitude smaller than what should be read internally for attention. We design AttAcc with multiple processing-in-memory devices, each multiplying the embedding vector with the portion of the KV matrices within the devices, saving external (inter-device) bandwidth and energy consumption.

**Index Terms**—Transformer-based generative model, processing-in-memory, attention.

## I. TRANSFORMER-BASED GENERATIVE MODELS

GENERATIVE artificial intelligence refers to machine learning models that can generate new content based on input context. Since its introduction, the Transformer-based generative model (TbGM) has demonstrated remarkable accuracy in various application types, including not only tasks in the field of natural language processing but also image and speech processing domains. Among TbGMs, Generative Pre-trained Transformers (GPTs) are gaining popularity due to their

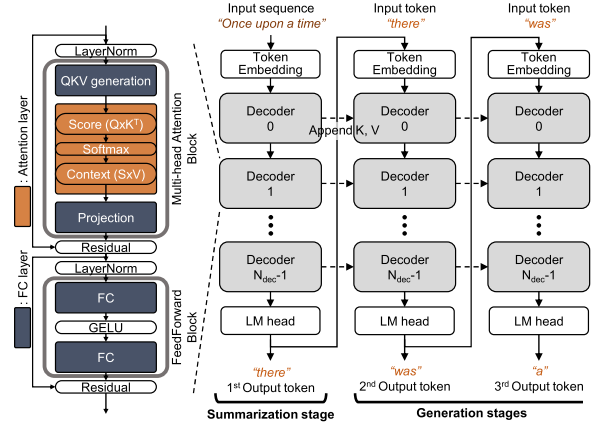


Fig. 1. The architecture and inference process of GPT.

outstanding text generation performance and the popularity of ChatGPT service.

Based on the observation that larger models and longer sequences can improve accuracy [7], [10], the size of pretrained weights has increased to hundreds of GB, and the recently published GPT-4 [5] has a maximum input sequence length of 32,768. Moreover, considering that the services such as ChatGPT can accumulate the previous context (e.g., conversation) of the same user and store multiple examples in the context window [1], TbGM is expected to have a long input sequence. The amount of computation and the memory working set size increase proportional to the model size and the sequence length. GPT-3 has up to 175 billion parameters and requires 1,475 TFLOPs of computation (i.e., more than  $300,000\times$  of ResNet-50) for inference with an input sequence length ( $L_{in}$ ) of 2,048 and an output sequence length ( $L_{out}$ ) of 2,048. As GPT-based services are recently drawing attention and the request traffic from their clients has surged rapidly, improving the serving throughput of TbGM inference is critical.

**GPT model:** The GPT architecture is based on the decoder of Transformer [8] (see Fig. 1). In a GPT model, input tokens are first converted into embedding vectors through Token Embedding and cascaded through  $N_{dec}$  decoders, each with different pretrained weights, generating an output embedding vector using these vectors as input. The output vector is converted to an output token through a language model (LM) head.

A decoder consists of a multi-head attention block and a feedforward layer. Both blocks are accompanied by layer normalization (LayerNorm), and residual, element-wise addition.

Manuscript received 6 June 2023; revised 23 July 2023; accepted 9 August 2023. Date of publication 15 August 2023; date of current version 19 September 2023. This work was supported in part by Institute of IITP grant funded by the Korea government (MSIT) under Grants 2021-0-00863 and IITP-2023-RS-2023-00256081 and in part by PRISM, one of the seven centers in JUMP 2.0, an SRC program sponsored by DARPA. (Corresponding author: Jung Ho Ahn.)

Jaewan Choi, Jaehyun Park, Kwanhee Kyung, and Jung Ho Ahn are with Seoul National University, Seoul 08826, South Korea (e-mail: jwchoi@scale.snu.ac.kr; jhpark@scale.snu.ac.kr; kwanhee.kyung@scale.snu.ac.kr; gajh@snu.ac.kr).

Nam Sung Kim is with the University of Illinois at Urbana-Champaign, Champaign, IL 61801 USA (e-mail: nam.sung.kim@gmail.com).

Digital Object Identifier 10.1109/LCA.2023.3305386

# Output tokens ( $L_{out}$ )	2,048	512	128	32	8	2
# Input tokens ( $L_{in}$ )	2	8	32	128	512	2,048
2,048	99.9	99.9	99.9	99.9	99.8	99.2
512	99.8	99.8	99.8	99.8	99.2	96.8
128	99.2	99.2	99.2	99.1	96.7	88.2
32	96.9	96.8	96.7	96.3	87.9	64.5
8	87.5	87.4	87.0	85.6	62.1	29.1
2	50.0	49.8	48.9	45.8	19.0	5.5

Fig. 2. Generation stage time ratio in total execution time for various numbers of input and output tokens on GPT-3.

The multi-head attention block is composed of three layers: the QKV generation layer, attention layer, and projection layer in order. QKV generation and projection layers are fully connected (FC) layers for generating query (Q), key (K), and value (V) data which contain the information of each input token and projecting the outputs of the attention layer. The attention layer calculates the contextual representation between tokens (Fig. 1 (left)). It operates as a multi-head; The Q, K, and V data are equally divided into  $N_{head}$  heads ( $Q_i$ ,  $K_i$ , and  $V_i$  for  $i = 0, 1, \dots, N_{head}-1$ ). Each head sequentially performs an inner product (score operation) on  $Q_i$  and  $K_i$ , a softmax operation on the result, and another inner product (context operation) with  $V_i$  on the softmax result. The outputs of each head are concatenated and used as input for the projection layer. The feedforward block consists of two FC layers and a Gaussian error linear unit (GELU) layer.

GPT inference of a request comprises a summarization (Sum) stage that understands the context of an input sequence, followed by multiple generation (Gen) stages that generate new tokens. Each stage proceeds sequentially and reuses the same pretrained weights. The primary operations in the Sum stage are the general matrix multiplication (GEMM). In the Sum stage, each decoder receives a matrix of  $L_{in} \times d_{emb}$  (dimension of embedding) as an input, processes it through FC and attention layers, and outputs a matrix of the same size. During the attention layer, the Sum stage generates key and value (KV) matrices with a size of  $2 \times L_{in} \times d_{emb}$ , containing the context of the input sequence.

The Gen stage proceeds similarly to the Sum stage but with two major differences. First, a Gen stage mainly performs the general matrix-vector multiplication (GEMV) operation as it takes a single token generated in the previous stage as input, and the input of each decoder is a vector. Second, the attention layer operates the Q vector for an input token of the current stage with the KV matrices that aggregate the input tokens of the Sum stage with the generated tokens up to the previous Gen stage. When  $L$  is defined as the sum of  $L_{in}$  and the number of tokens generated up to the current stage, the dimensions of the Q vector and the KV matrix are  $1 \times d_{emb}$  and  $L \times d_{emb}$ , respectively. The output token of a Gen stage becomes the input of the next Gen stage, being repeated until an end-of-sequence token is generated.

The Gen stages typically overwhelm the Sum stage in execution time due to their sequential nature of reading the entire pretrained weights per token. Fig. 2 shows the portion of time taken by the Gen stages over the total execution time for various  $L_{in}$  and  $L_{out}$  in the GPT-3 175B model. We experimented with our in-house simulator, assuming the peak FLOPs (2.5 PFLOPs) and maximum memory bandwidth (16 TB/s) of the NVIDIA

DGX A100 640 GB (DGX). When both  $L_{in}$  and  $L_{out}$  are 32, the portion of Gen stages exceeds 96%. Even if  $L_{in}$  is 2048, and  $L_{out}$  is 128, that of Gen stages also exceeds 88%. This dominance can also be confirmed in the prior works [2], [11] dealing with GPT-2. We focus on the Gen stages hereafter.

## II. ANALYSIS OF BATCHING FOR TbGM

Batching has been widely used to improve the serving throughput of machine learning inference, which is a methodology that processes multiple client requests for the same model in the form of grouped tasks. In the case of TbGM, we assume stage-level scheduling for batching as in [9] considering that the number of stages (same as  $L_{out}$ ) is different for each request. That is, when a request in a batch is completed, a new request is added to the batch in the next stage instead of waiting for the other requests in a batch to be completed.

### A. Benefits of Batching for TbGM

Applying the batching in GPT improves the reusability of weights in computing the FC layer; it improves both the throughput and energy efficiency of conventional serving platforms, such as DGX. When batched, the FC layer sharing the weights across different requests becomes a GEMM operation instead of a GEMV operation. This means that once weights are read from the off-chip memory, all requests in the batch can be processed simultaneously, improving the Op/B (the number of arithmetic operations per byte accessed from off-chip memory) of the FC layer. Consequently, there is little change in the execution time of the FC layer, and the throughput of FC layers improves by a factor equivalent to the batch size. Moreover, off-chip memory accesses to the weights decrease proportionally to the batch size, reducing the energy consumption of the off-chip memory. Fig. 3(a) and (b) represent the throughput (generated tokens per second) and energy consumption for off-chip memory accesses. For all three configurations of  $L_{out}$  with  $L_{in}$  of 2,048, increasing the batch size improves throughput and reduces energy consumption.

### B. Limitations of Conventional Platforms for Batching

Despite the benefits of batching, new challenges due to the attention layer arise as the batch size increases with the trend of long input sequences; they impose limitations on conventional platforms. First, the size of the KV matrices, which grows with batch size, could surpass the memory capacity of conventional platforms. The attention layer has different KV matrices for each request, with their sizes being proportional to the sequence length. The number of elements of KV matrices is  $(L_{in} + L_{out}) \times d_{emb}$  per decoder. For example, on the GPT-3 175B model using FP16 with both  $L_{in}$  and  $L_{out}$  of 2048, the size of KV matrices is 18 GB per request; thus, the required memory capacity for a batch size of 64 is about 1.5 TB, including 326 GB of the weights and 1,152 GB of KV matrices, greatly surpassing DGX's 640 GB memory capacity (see Fig. 3(a)).

Second, the latency of executing the attention layer increases with batching, and with large batch sizes, it takes a large portion of the execution time. The latency of executing an FC layer is

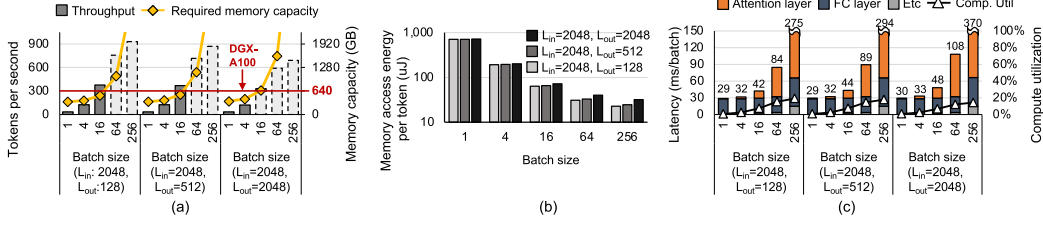


Fig. 3. (a) Throughput, required memory capacity, (b) memory access energy consumption per output token, and (c) the Gen stage time breakdown and GPU compute utilization for various batch sizes with  $L_{in}$  of 2048 and  $L_{out}$  of 128, 512, or 2048 when running the GPT-3 175B model on DGX assuming unlimited memory capacity. The dotted bars in (a) indicate the throughput with a batch size not available with DGX A100 (640 GB) due to memory capacity limitation.

nearly uniform regardless of batch sizes until the Op/B ratio in the course of computing the FC layer exceeds that provided by DGX (when the batch size is 146). In contrast, the latency of executing the attention layer increases steadily with batch size, accounting for more than 60 percent of execution time when the batch size is 64 (see Fig. 3(c)).

The attention layer has a low arithmetic intensity regardless of batch size. The attention layer in the Gen stage primarily performs GEMV between the Q vector generated by the input token and the KV matrices, exhibiting a low Op/B ( $\sim 1$ ). Unlike the FC layer, the attention layer still has memory-intensive GEMV operations even after batching due to the sequential nature of processing the Gen stages. Therefore, the computing units of DGX are mostly idle for the attention layer, even with batching. As shown in Fig. 3(c), the GPU utilization increases due to the improved Op/B of the FC layers. However, even when the batch size reaches 256, the utilization remains below 20% because of the substantial time spent in the attention layer.

Moreover, the increased latency due to the attention layer could potentially violate a Service Level Objective (SLO). If the SLO of GPT-3 is set to 50 ms per output token, within a reasonable range, a Gen stage with a batch size of 64 and  $L_{in}/L_{out}$  of 2048 would violate the SLO as the latency rises to 108 ms (see Fig. 3(c)), necessitating a limit on the batch size to 16 or fewer to meet the SLO. Thus, if considering the SLO, conventional platforms need to further increase expensive off-chip memory bandwidth for attention layer acceleration to process large batches.

### III. TBGM ACCELERATOR ARCHITECTURE

To tackle the inefficiency of conventional platforms executing TbGM, we propose a distributed heterogeneous computing system. It consists of high-performance compute units (xPUs), such as GPUs or TPUs, and AttAcc, which are connected by a commercial high-bandwidth interconnect, such as PCIe, NVLink, and CXL. This system accelerates the low Op/B attention layers with AttAcc, while executing the high Op/B FC layers with large batch sizes on xPUs.

Focusing on AttAcc in this paper, we start the design with a PIM (processing-in-memory) architecture adopted by HBM-PIM [3], which directly couples processing elements (PEs) with each DRAM bank at the I/O boundary and activates  $N_{PE}$  such DRAM banks in parallel to expose  $N_{PE} \times$  higher internal bandwidth to the PEs than external bandwidth to the xPUs. Such

a PIM is compelling for a baseline AttAcc due to the following two reasons.

First, KV matrices are written once at the Sum stage but used many times at the Gen stages. Such a write at the Sum stage and the reuses of the KV matrices at the Gen stages require the external bandwidth and the internal PIM bandwidth, respectively. Considering that writing KV matrices at the Sum stage of a batch can be amortized to reading at the Gen stages of the previous batch, PIM's external bandwidth requirement for KV matrices could be reduced by orders of magnitude.

Second, the GEMV operations of the attention layer during the Gen stages have much smaller inputs and outputs than the large KV matrices to process. In the attention layer, the score and context operations, each performing  $N_{head}$  GEMVs, have the sizes of (input, K or V matrix, output) as  $(1 \times d_{emb}, L \times d_{emb}, N_{head} \times L)$  and  $(N_{head} \times L, L \times d_{emb}, 1 \times d_{emb})$ . Thus, for the GEMVs of the attention layer, the size ratio of the input and output data over the KV matrices corresponds to  $(d_{emb} + N_{head} \times L) / (L \times d_{emb})$ . For GPT-3 with 175B weights,  $d_{emb}$  and  $N_{head}$  are 12,288 and 96, respectively. Assuming a large  $L$  (e.g.,  $> 2048$ ), this ratio is about  $N_{head}/d_{emb} = 1/128$ , implying the ratio of PIM's external and internal bandwidth requirements.

AttAcc has an ample design space to explore. It includes what to use for baseline memory type, where to place PEs for GEMV and softmax operations within the DRAM organization hierarchy, whether to use 3D-stacking or chiplet integration technology if we place controllers and expensive PEs on different dies, whether to use a logic process or a DRAM process for those dies, and how to map the attention layer to the DRAM banks (similar to [6]). While leaving such deep design space explorations as future work, we focus on HBM-PIM-based AttAcc and propose to place the PEs for GEMV on the DRAM die and place the PEs for expensive operations such as exponentiation for softmax operation on the HBM buffer die.

We map each attention head to one HBM. Each bank processes a portion of  $K_i$  and  $V_i$  divided into column-wise and row-wise, respectively. As GEMV operations in the attention layer have an enough degree of parallelism across multiple requests, heads, and tokens, bank-level parallelism can be fully utilized.

AttAcc is generally not appropriate for accelerating TbGM training because it mostly proceeds with compute-intensive Sum stages. Instead, AttAcc can be utilized for fine-tuning from human preferences, such as Reinforcement Learning from Human Feedback, which entails a Gen stage.



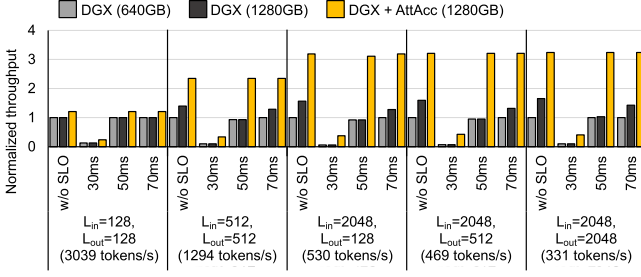


Fig. 4. The normalized throughput of *DGX(640 GB)*, *DGX(1280 GB)*, and *DGX+AttAcc(1280 GB)* for various SLOs. The baseline is *DGX(640 GB)* without the SLO, with absolute values in parentheses.

#### IV. EVALUATION

**Experimental setup:** We compared a system combining *DGX* and *AttAcc* (*DGX+AttAcc*) with a baseline *DGX*-only system (*DGX(640 GB)*) and the baseline but with twice the memory capacity (*DGX(1280 GB)*). The memory capacity of *DGX+AttAcc* is 1280 GB in total where that of *DGX* is 326 GB to store the weights and that of *AttAcc* corresponds to the rest. The aggregate internal bandwidth of *AttAcc* is 64 TB/s,  $4\times$  higher than that of *DGX* [3]. The energy per bit transfer of HBM for PIM operations is assumed to be  $3.5\times$  less than that of *DGX*'s HBM [3].

We developed an in-house simulator to measure the performance and energy efficiency of the systems. The simulator receives *DGX* and *AttAcc* configurations as well as model information as input and outputs the execution time and energy consumption of each system. We used the DRAM access energy as that of HBM2 reported in [4].

We set GPT-3 175B as the target model with the data type of FP16. We used the average sequence length of requests in a batch for various configurations of  $L_{in}$  and  $L_{out}$  assuming that enough requests can be batched. The SLO of TbGMs is different for each service, and there is no specific information disclosed. Therefore, we inferred approximate values of the latency target per token by referring to [9] and explored the scenario without the SLO and with 30 ms, 50 ms, and 70 ms latency targets.

**Performance:** *DGX+AttAcc* achieves up to  $3.24\times$  and  $2.04\times$  higher throughput for *DGX(640 GB)* and *DGX(1280 GB)*, with greater  $L_{in}$  leading to larger throughput improvement (see Fig. 4). This improvement attributes to *DGX+AttAcc*'s increased memory capacity to support larger batch sizes than *DGX(640 GB)* and its use of the PIM architecture that provides higher internal aggregate memory bandwidth for the attention layer. Considering the SLO, *DGX+AttAcc* can achieve further throughput improvement than *DGX(640 GB)* and *DGX(1280 GB)*, which have limited batch size due to the latency of the attention layer (e.g.,  $5.86\times$  than *DGX(1280 GB)* with SLO of 30 ms). Overall, the tighter the SLO, the greater the throughput improvement of *DGX+AttAcc*.

**Energy consumption:** *DGX+AttAcc* achieves higher energy efficiency for off-chip memory access than *DGX* systems (see Fig. 5). First, due to the larger batch size, the weight reusability of FC layers increases, reducing the number of off-chip memory accesses. Second, the energy consumption of the attention layer is reduced because of the memory access energy reduction of

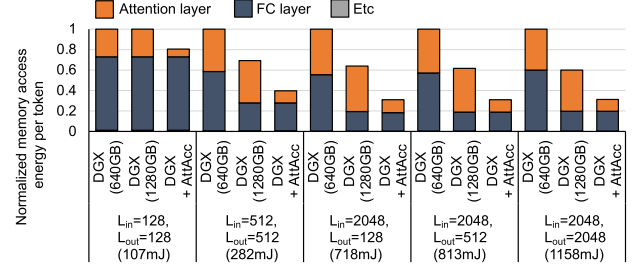


Fig. 5. Normalized memory access energy per output token of *DGX(640 GB)*, *DGX(1280 GB)*, and *DGX+AttAcc* for GPT-3 175B without SLO. The baseline is *DGX(640 GB)*, with absolute values in parentheses.

the PIM architecture. Without considering the SLO, memory access energy consumption is reduced by up to 69% and 51% compared to *DGX(640 GB)* and *DGX(1280 GB)*, respectively.

#### V. CONCLUSION

We have analyzed the impact of batching for Transformer-based generative models and identified the growing importance of the attention layer in the trend with increasing model sizes and sequence length. The conventional serving platforms, such as GPUs, are suboptimal for large batch sizes having stringent memory capacity and bandwidth requirements processing the attention layer under a tight service-level objective. We proposed *AttAcc*, an accelerator for the attention layer, composed of cost-effective processing-in-memory devices exploiting the characteristic that the external bandwidth requirement is far less than the internal bandwidth requirement for the attention layer. Compared to the monolithic state-of-the-art GPU system, the heterogeneous system combining GPUs with *AttAcc* achieved significantly higher throughput (up to  $3.24\times$ ) and energy efficiency (up to  $3.22\times$ ).

#### REFERENCES

- [1] T. Brown et al., "Language models are few-shot learners," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.
- [2] S. Hong et al., "DFX: A low-latency multi-FPGA appliance for accelerating transformer-based text generation," in *Proc. IEEE/ACM 55th Int. Symp. Microarchitecture*, 2022, pp. 616–630.
- [3] S. Lee et al., "Hardware architecture and software stack for PIM based on commercial DRAM technology: Industrial product," in *Proc. IEEE/ACM 48th Annu. Int. Symp. Comput. Architecture*, 2021, pp. 43–56.
- [4] M. O'Connor et al., "Fine-grained DRAM: Energy-efficient DRAM for extreme bandwidth systems," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, 2017, pp. 41–54.
- [5] OpenAI, "GPT-4 technical report," 2023. [Online]. Available: <https://arxiv.org/abs/2303.08774>
- [6] J. Park et al., "TRiM: Enhancing processor-memory interfaces with scalable tensor reduction in memory," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, 2021, pp. 268–281.
- [7] M. Shoybi et al., "Megatron-LM: Training multi-billion parameter language models using model parallelism," 2019, *arXiv: 1909.08053*.
- [8] A. Vaswani et al., "Attention is all you need," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
- [9] G.-I. Yu et al., "ORCA: A distributed serving system for transformer-based generative models," in *Proc. 16th USENIX Symp. Operating Syst. Des. Implementation*, 2022, pp. 521–538.
- [10] M. Zaheer et al., "Big bird: Transformers for longer sequences," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 17283–17297.
- [11] M. Zhou et al., "TransPIM: A memory-based acceleration via software-hardware co-design for transformer," in *Proc. IEEE Int. Symp. High-Perform. Comput. Architecture*, 2022, pp. 1071–1085.