전자통신동향분석, 40권 6호 2025, 138-146

PIM 기반 LLM 추론 가속 연구 동향

Research Trend on PIM-based LLM Inference Acceleration

안백송 (B.S. An, bsahn@etri.re.kr) 고성능컴퓨팅시스템연구실 책임연구원
김민서 (M.S. Kim, kms0926@sogang.ac.kr) 서강대학교 고성능인공지능시스템 연구실/연구원
김주영 (J.Y. Kim, ephes413@sogang.ac.kr) 서강대학교 고성능인공지능시스템 연구실/연구원
한석기 (S.G. Han, hski0930@sogang.ac.kr) 서강대학교 고성능인공지능시스템 연구실/연구원
이영민 (Y.M. Yi, ymyi@sogang.ac.kr) 서강대학교 인공지능학과/교수

ABSTRACT

Following the rapid expansion of large-language-model (LLM) applications, memory bottlenecks in inference have emerged as key challenges limiting performance and efficiency. Conventional GPU-based accelerators excel at computationally intensive operations (e.g., GEMM) but suffer from severe underutilization in memory-intensive operations, such as KV cache access and batched GEMV. To address this issue, processing-in-memory (PIM) has attracted increasing attention, and numerous PIM-based approaches for accelerating LLM inferences have recently been proposed. This paper surveys state-of-the-art studies, including NeuPIMs, IANUS, AttAcc, Höppen, and PAPI, and analyzes architectural designs, scheduling techniques, and memory-access optimizations for PIM-based inference acceleration. NeuPIMs and IANUS integrate or parallelize NPU and PIM to improve resource utilization, whereas AttAcc uses an HBM-based design that performs all attention computations inside the memory. These studies highlighted the limitations of PIM and presented an alternative GPU–NPU hybrid architecture with separate memory, which introduced heterogeneous PIM accelerators to handle the dynamic workload characteristics in the decoding stage. Collectively, these studies demonstrate diverse strategies to mitigate memory bottlenecks and enhance energy efficiency in LLM inference, suggesting that PIM will likely evolve into a core technology for accelerating LLM inference within heterogeneous systems that combine GPU and NPU resources.

KEYWORDS Heterogeneous Accelerator, Inference, Large Language Model(LLM), Processing-In-Memory(PIM)

* DOI: https://doi.org/10.22648/ETRI.2025.J.400613
* 이 연구는 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임[No.RS-2025-02263167, AI 반도체 통합 자원관리 기술 개발].

I. 서론

최근 LLM(Large Language Model)의 성능이 다양한 태스크에 적용될 수 있을 정도로 향상됨에 따라 LLM 추론의 수요가 폭발적으로 증가하고 있다. 이에 따라 효율적인 LLM 추론 필요성이 갈수록 커지고 있다. LLM 추론은 학습과 달리 단일 사용자(싱글 배치) 또는 다중 사용자라고 하더라도 수십을 넘기가 힘들기 때문에 4, 8, 16, 32 등에서 결정되는 경우가 많으며, 메모리 병목이다. LLM은 동일한 트랜스포머 레이어를 적층하여 구성되는데, 트랜스포머는 크게 Q, K, V 행렬을 생성하는 선형 변환(Linear Projection), 토큰 간의 관계를 파악하는 셀프 어텐션(Self-Attention), 토큰의 특징 표현을 풍부하게 만드는 FFN(Feed Forward Network)의 세 단계로 구분할 수 있는데, KV(Key Value) 캐시에 접근하고 Batched GEMV(GEneral Matrix Vector Multiplication)으로 수행되는 셀프 어텐션이 가장 메모리 병목이고, 나머지 두 단계도 배치 크기가 작을 경우 GEMV와 비슷한 수준의 GEMM(GEneral Matrix Matrix Multiplication)이라서 메모리 병목이다.

이를 해결하기 위해 FlashAttention[1]처럼 셀프 어텐션의 전역 메모리 사용량과 접근을 줄이는 방법도 제안되었고, Speculative Decoding[2]과 같이 여러 토큰을 병렬적으로 처리함으로써 배치 크기를 키워 메모리 병목을 해소하는 방법도 제안되었으며, HBM(High Bandwidth Memory)처럼 빠른 메모리를 장착하는 하드웨어적 방법이 널리 사용되고 있다. PIM(Processing In Memory)은 메모리에 연산기능을 갖춘 장치로서, 최근 PIM을 이용하여 LLM 추론의 병목을 해소하려는 노력들이 활발히 이루어지고 있다. 본고에서는 이런 접근법을 따르는 일련의 최근 논문들을 분석하고 동향을 파악한다.

II. 배경

1. LLM 추론의 연산 특성

토큰 간의 관계를 파악하는 셀프 어텐션을 수행하기 위해서는 Query(Q), Key(K), Value(V)의 세 가지 다른 표현이 필요한데, 선형 변환(Linear Projection)은 이를 얻기 위해, 입력 벡터 또는 행렬 X와 학습을 통해 구한 WQ, WK, WV 가중치 행렬을 곱하여 입력을 선형적으로 각각 변환한다. 단일 배치 추론일 경우 GEMV에 연산에 해당하며, 다중 배치 추론일 경우도 배치 크기가 크지 않기 때문에 여전히 메모리 병목이다. 셀프 어텐션은 대개 여러 헤드를 가진 MHA(Multi-Head Attention) 형태인데 각 헤드마다 상대적으로 작은 개별적인 가중치 행렬을 가지지만, 구현상 이를 하나의 큰 행렬로 만들어 행렬곱셈을 수행하므로, 멀티 헤드라고 해서 메모리 병목이 더 가중되지는 않는다.

셀프 어텐션은 K 전치행렬과 Q 행렬을 곱한 후 스케일 후에 이에 대한 Softmax를 통해 얻어진 행렬 A에 V 행렬을 곱하여 출력 행렬, 즉 컨텍스트 행렬 O를 구한다. 이때, 추론은 단일 배치일 경우 q와 a는 모두 벡터가 되어, 2개의 GEMV 연산이라고 할 수 있다. K와 V는 기존 토큰들의 key와 value를 담고 있는 행렬로서 KV 캐시에서 읽어온다. 또한, 선형변환처럼 가중치 행렬을 합칠 수 없어서, Batched GEMV로 수행해야 한다. 따라서, 셀프 어텐션이 메모리 병목이 가장 심하다.

FFN의 경우, 크게 Gate, Up, Down에 해당하는 3개의 가중치 행렬과 컨텍스트 벡터를 곱하는 3개의 GEMV라고 할 수 있다. 단일 배치가 아니라 다중 배치가 된다고 해도, 그 크기가 작을 때는 GEMM이지만 성능상 GEMV와 크게 다를 바 없다.

2. PIM

프로세싱 코어 또는 ALU(Arithmetic Logic Unit) 장치의 개수가 많아질수록 ALU 연산은 가속이 잘 되지만, 메모리 접근시간의 단축이나 대역폭의 증가는 상대적으로 프로세싱 코어 수의 증가보다 더디기 때문에 GPU(Graphics Processing Unit)와 같은 매니코어 아키텍처에서는 메모리 접근이 상대적으로 비싼 연산이 되며 메모리 접근이 많은 응용은 GPU의 코어 개수가 증가할수록 오히려 더 성능 병목이 될 수 있다. 메모리 병목이 발생하는 근본적인 이유는 GPU를 비롯한 일반적인 프로세서들이 계산장치와 메모리가 구분된 폰 노이만 구조를 가지기 때문인데, PIM(Processing-In-Memory)은 메모리 소자 내 혹은 근처에 계산 회로를 추가하여 데이터 전송 없이 메모리 내부에서 직접 연산을 수행할 수 있는 메모리 및 기술을 일컫는다. 대표적으로, DRAM(Dynamic Random Access Memory) 안에 비트 연산이 가능한 연산회로를 추가한 Ambit[3]에서 DRAM 내부 회로를 활용한 PIM의 효용성에 대해 보였고, NeuroPIM[4]에서 딥러닝 가속을 위해서 PIM이 활용될 수 있음을 보였다. 또한, 상용적으로 SK하이닉스에서는 AiM(Accelerator in Memory)[5]이라는 PIM을 출시하였으며, 삼성전자도 HBM-PIM[6]을 출시하였다. UPMEM[7]의 경우 DRAM 내 작은 RISC 코어를 내장하여 제한된 연산을 지원하는 방식으로 PIM을 구현하였다.

III. PIM 기반 LLM 추론의 가속

본 논문에서는 메모리 병목인 LLM 추론을 효율적으로 가속하기 위해 PIM을 사용한 최신 일련의 연구들을 살펴보고, 그 연구 동향을 분석한다. 먼저 ASPLOS 2024에서 발표된 세 편의 논문들을 살펴본다: KAIST 등에서 발표한 NeuPIMs[8]은 하드웨어적으로는 메모리 뱅크에 일반적인 설계와 달리 2개의 행을 둠으로써, NPU(Neural Processing Unit)로부터 메모리 접근과 PIM의 GEMV를 동시에 병렬적으로 수행할 수 있게 했다는 점, 스케줄링 측면에서 하나의 큰 배치를 두 개의 서브배치로 나누어 NPU와 PIM의 자원배치를 효율적으로 했다는 점이 특징이다. 서울대학교와 SK하이닉스, 사피온, KAIST 등에서 발표한 IANUS[9]는 NPU와 PIM을 위한 통합 메모리를 가정하여 데이터 중복을 피했고, 이때 NPU의 메모리 접근과 PIM 연산의 충돌을 피하기 위한 스케줄링 정책을 제안하였으며, AiM과 사피온 NPU를 사용하여 FPGA(Field Programmable Gate Array) 프로토타입으로 검증하였다. AttAcc[10]은 서울대와 UIUC에서 발표한 연구로서, HBM에서 어텐션 스코어뿐만 아니라 어텐션 블록의 전체 연산을 수행할 수 있도록 하드웨어를 제안하였다.

다음으로, ISCA 2025와 ASPLOS 2025에서 발표된 두 편의 논문을 살펴본다: Hybe[11]는 KAIST에서 발표한 논문으로서, PIM과는 무관하지만 PIM의 한계와 단점을 강조하면서, 분리된 메모리를 가지는 GPU와 NPU가 더 효과적이라고 주장한다. PAPI[12]는 Speculative Decoding에서 Speculation 길이, 즉 드래프트 모델이 몇 개의 토큰까지 생성할지에 따라, 타겟 모델의 배치 크기가 달라지는 점 등에 착안하여 메모리 병목에 더 적합한 PIM 가속기와 보다 계산 집약적인 PIM 가속기 두 종류의 PIM 가속기를 제안하였다.

1. NeuPIMs

LLM 추론은 연산 집약적인 GEMM과 메모리 집약적인 GEMV가 교차적으로 요구되며, 단일 가

속기로는 이질적인 특성을 모두 충족하기 어렵다. NeuPIMs는 이를 해결하기 위해 GEMM에 최적화된 NPU와 GEMV에 특화된 PIM을 결합한 이기종 가속 시스템으로, 기존 PIM의 차단(Blocked) 모드 한계를 극복하기 위해 각 뱅크에 이중 행 버퍼를 도입해 메모리 접근과 PIM 명령을 동시에 처리한다. 또한, GEMM과 GEMV 간의 의존성으로 인해 병렬 처리가 제한되는 문제를 완화하기 위해, 런타임 서브 배치 인터리빙 기법을 적용해 두 개의 서브 배치를 교차 실행하고 파이프라인으로 처리함으로써 동시 실행을 극대화한다.

1.1 Motivation

LLM 추론은 입력을 인코딩하는 Prefill 단계와 토큰을 순차 생성하는 Decoding 단계로 나뉘며, 두 단계 모두 QKV(Query Key Value) 생성, 멀티헤드 어텐션(MHA), 피드포워드 네트워크(FFN)로 구성된다. QKV와 FFN은 본래 GEMV 연산이지만 요약화 단계나 배치 상황에서는 GEMM으로 변환되고, MHA는 배치가 불가능해 항상 GEMV로 남는다. 그 결과 요약화 단계는 연산 병목, 생성 단계는 메모리 병목에 묶여 GPU 기반 시스템은 메모리 용량은 가득 차더라도 연산 활용률이 40% 이하로 떨어진다. 이를 보완하기 위해 제안된 NPU-PIM 통합 방식 역시 메모리 뱅크가 단일 행 버퍼를 사용하기 때문에 메모리 읽기/쓰기 연산과 GEMV 전용 PIM 가속을 동시에 수행하지 못하고, 두 가속기의 결합 활용도가 40% 미만에 그친다. 따라서 NPU와 PIM의 병렬 실행을 가능하게 하는 것이 핵심 과제로 남아 있다.

1.2 Approaches

NeuPIMs는 이러한 한계를 해결하기 위해 메모리 뱅크에 두 개의 행 버퍼(MEM 행 버퍼, PIM 행 버퍼)를 갖추어 각각을 독립적인 데이터 경로에 연결한다. 이러한 분리 덕분에 메모리 읽기/쓰기 연산과 PIM의 GEMV 연산을 동시에 수행할 수 있으며, 마이크로아키텍처적 복잡성은 최소화한 채 명령어 인터페이스와 메모리 제어 메커니즘으로 제어된다. 이중 행 버퍼 구조는 멀티헤드 어텐션 연산을 헤드 단위로 분해할 때, PIM에서 수행되는 로짓 및 어텐드 연산과 NPU 벡터 유닛에서 수행되는 소프트맥스 연산을 전체 결과를 기다리지 않고 부분적으로 중첩 실행할 수 있게 해 자원 활용도를 높인다.

또한, 단순한 NPU-PIM 통합 장치에서는 QKV 생성, 멀티헤드 어텐션, 프로젝션 및 FFN 간의 의존성으로 인해 직렬 실행이 불가피했지만, NeuPIMs는 하나의 큰 배치를 두 개의 서브 배치로 분할해 교차 실행하는 서브 배치 인터리빙 기법을 도입한다. 아울러 채널 부하 균형 알고리즘으로 멀티헤드 어텐션의 토큰 길이를 균등하게 분배해 긴 시퀀스 처리에 따른 지연 시간을 줄이고, 서브 배치 분할 알고리즘으로 두 서브 배치의 실행 시간을 유사하게 맞추어 병렬 실행의 효율을 극대화한다.

GPU-only 대비 3배 이상 처리속도를 향상하였고, NPU-only에 비해서는 2.4배, 단순 NPU+PIM 대비 1.6배 향상하였다.

2. IANUS

IANUS(Integrated Accelerator based on NPU-PIM Unified Memory System)는 엔드 투 엔드 LLM 추론의 다양한 연산을 가속하기 위한 NPU-PIM 통합 메모리 시스템 기반 아키텍처이다. 해당 시스템에서는 PIM 연산과 메모리 접근이 동시에 수행될 수 없다는 문제가 발생한다.

본 연구는 작업 부하 매핑과 명령어 스케줄링을 포함하는 PIM Access Scheduling(PAS)을 도입해 자원 충돌과 데이터 의존성을 해결했다. 아키텍처의 실

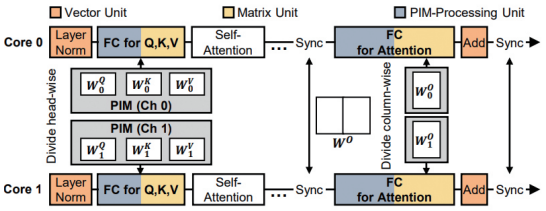
현 가능성을 검증하기 위해 상용 PIM(AiM), NPU(S-APEON-X330)과 FPGA(VCU118)를 활용하여 프로토타입을 설계했다.

2.1 Motivation

Transformer 기반 모델은 널리 사용되지만, 연산 특성이 다양해서 엔드 투 엔드 추론 가속이 어렵다. xPU는 연산 집약적 DNN(Deep Neural Network) 연산에는 강하지만, 메모리 병목이 발생하는 연산에서는 성능이 제한된다. 반면에 PIM은 메모리 근처에서 연산을 수행하여 데이터 이동을 최소화하므로 메모리 병목인 연산에 강하다. 기존의 연구는 이러한 강점을 활용하기 위해 주로 어텐션 레이어에서 PIM을 이용한 추론 가속에 집중했다. 또한, 이기종 가속기 환경에서도 분할 메모리 시스템을 적용한 기존의 연구는 독립된 메모리 공간에 복제된 데이터를 두 배로 저장해야 한다는 문제로 인해 LLM에는 적합하지 않은 경향을 보였다. 따라서 엔드 투 엔드 추론 가속을 위해서는 다양한 연산 집약도를 갖는 연산을 효율적으로 처리할 수 있으면서도 메모리 사용량을 최소화할 수 있는 이기종 가속기 통합 하드웨어가 필요했다.

2.2 Approaches

IANUS는 Transformer 기반 모델의 특정 연산에 대해 최적화된 가속기로 처리하는 방식에서 벗어나 엔드 투 엔드 추론 가속을 위해 NPU-PIM 통합 메모리 시스템을 제안한다. 이처럼 NPU와 PIM이 같은 메모리를 공유하도록 하여 데이터 사용량 및 이동 비용을 감소시켰다. 특히 공유하는 메모리에 대한 의존성을 파악하고 PIM 연산을 매크로화하여 메모리 접근 시의 의존성이 보존되도록 설계했다. 본 연구에서 FC(Fully Connected) 레이어에 대해 요약 단계에서는 연산 집약적인 GEMM 연산이므로



출처 Reprinted from M.S. Seo et al., "Ianus: Integrated accelerator based on NPU-PIM unified memory system," in Proc. ACM Int. Conf. Archit. Support Program. Lang. Oper. Syst., (La Jolla, CA, USA), Apr. 2024, pp. 545-560.

그림 1 IANUS의 NPU와 PIM에 대한 연산 할당 및 병렬 수행

NPU에서 처리하도록 한다. 반면에 생성 단계에서는 연산 집약도가 달라짐을 확인하였고, 명령어 단위로 NPU의 MU(Matrix Unit)와 PIM에서의 처리 시간을 비교하여 더 짧은 지연을 갖는 가속기에서 처리된다(그림 1). 어텐션 레이어에 대해 Score 및 Context 연산은 GEMV 연산에 해당하여 PIM에서 처리 가능하지만, Head의 차원이 작은 환경에서는 PIM의 병렬성 활용이 낮아지는 점에 근거하여 MU에서 처리, Softmax는 벡터 연산에 해당하므로 VU(Vector Unit)에서 처리하도록 했다. GPT-2 시뮬레이션 결과 IANUS는 NVIDIA A100 대비 6.2배, 최신 FPGA 기반 가속기(DFX) 대비 3.2배의 성능 향상을 보였다.

3. AttAcc

AttAcc(Attention Accelerators)은 트랜스포머 기반 모델의 추론에서 어텐션 레이어 연산 시 발생하는 메모리 병목 문제를 해결하기 위한 PIM 기반 아키텍처이다. 해당 연구에서는 배치 크기 증가에 따라 연산 집약도가 높아지는 FC 레이어, 즉 FFN 블록은 xPU가 처리하고, 여전히 메모리 집약적인 연산이 포함된 어텐션 레이어는 PIM에서 처리하는 이기종 가속기(xPU+PIM)를 활용한 추론 방식을 시도했다. 이에 따라 SLO(Service Level Objective)에 의해

배치 크기가 제한되었던 경우, 이를 증가시킴으로써 GPU의 FC 레이어 처리속도를 크게 향상시킬 수 있었다.

3.1 Motivation

기존에 메모리 집약적인 연산을 PIM에서 처리하려는 연구는 지속되고 있었지만, 배치 사이즈를 고려한 연구는 거의 진행되지 않았다. 특히 어텐션 스코어 연산만 PIM에서 처리하던 기존 연구와 달리 AttAcc에서는 어텐션 레이어 전체를 PIM에서 처리하고자 시도했다. 이를 위해 각 연산의 성격에 맞춰 GEMV 유닛, Softmax 유닛, Accumulator 유닛을 HBM 내에 배치한 새로운 아키텍처를 설계했다.

3.2 Approaches

어텐션 레이어에서의 연산은 qKT에 해당하는 스코어(Score), 스케일링 후 소프트맥스를 취하는 소프트맥스(Softmax), 이 값에 V를 곱하는 컨텍스트(Context)의 세부적인 세 단계로 이루어져 있다. Score, Context는 가중치 행렬과 입력 벡터의 곱셈(GEMV)으로 이루어져 있다. AttAcc는 이러한 특성을 고려하여여 HBM 내부에 뱅크 단위의 GEMV 유닛과 Bank Group(BG) 단위의 Accumulator 유닛을 배치하고, Softmax는 별도의 유닛을 Buffer Die에 배치한다. 이를 통해 연산 과정에서의 데이터 이동을 최소화했다. 어텐션 레이어를 구성하는 세 연산이 차례대로 다른 유닛에서 수행되는 점을 이용한 파이프라이닝과 어텐션 레이어 이후의 FFN 레이어에서 AttAcc가 유휴 상태라는 점을 이용한 FFN 레이어의 동시 처리기법도 제안했다. 해당 연구에서는 Ramulator[13,14]를 수정한 자체 시뮬레이터를 통해 각 유닛의 배치 방식과 파이프라이닝 방식에 대해 처리 시간과 에너지 효율의 관점에서 평가했다. 이를 통해 AttAcc의 효율성을 극대화하여 같은 메모리 용량을 갖는 GPU 시스템 대비 처리율은 최대 2.81배 증가시키고, 에너지 효율성은 최대 2.67배 증가시킬 수 있음을 실험적으로 보였다.

4. Hybe

Hybe는 추론 단계에 따라 가속기를 분리해 Prefill 단계에는 기존의 GPU를 사용하고 Decode 단계에는 경량화한 NPU 아키텍처를 사용했다. 또한, GPU의 KV 캐시 메모리가 부족한 문제를 해결하기 위해 GPU에서 KV 캐시를 생성하고 NPU 메모리 형태로 Reshaping한 후, 실시간으로 NPU로 전송하는 Fine-grained KV Transmission을 구현했다. 그리고 오버로딩과 오프로딩 기술로 GPU와 NPU의 유휴 시간을 제거하고 활용도를 극대화하기 위해 단계 단위 파이프라이닝(Stage-Wise Pipelining)을 도입했다.

4.1 Motivation

모델이 일관성 있고 문맥에 맞는 답변을 생성하게 하려면 Context window 크기, 즉 시퀀스 길이를 늘이는 것이 필요한데, 이는 막대한 계산 자원을 요구하여, 구현에 어려움이 있다. 특히 문제가 되는 부분이 KV 캐시이다. Context Window가 커지면 KV 캐시의 메모리 점유량이 모델 파라미터 크기를 초과할 정도로 커진다. 이는 기존의 GPU가 추론 시 활용도를 높이기 위해 사용하던 배칭(Batching) 기법을 비효율적으로 만든다. 또한, Context Window 크기가 커짐에 따라 KV 캐시가 커지면서 PIM의 효용성도 떨어질 수 있는데, 일반 메모리 접근과 PIM 연산이 충돌할 가능성이 높아진다.

LLM 추론 과정은 계산 집약적인 Prefill 단계와 메모리 집약적인 Decode 단계로 나뉘는데, GPU는

Prefill 단계에서는 높은 성능을 보이지만 Decode 단계에서는 하드웨어 자원활용이 현저히 낮은 문제를 보였다. 따라서 두 단계를 모두 효율적으로 처리할 새로운 하이브리드 시스템이 필요했다.

4.2 Approaches

Hybe는 연산 특성이 다른 두 추론 단계를 각기 다른 가속기에 할당하는 방식으로 접근했다. 계산 집약적인 Prefill 단계(GEMM 위주)는 GPU가 처리하고, 메모리 병목이 발생하는 Decode 단계(GEMV 위주)는 맞춤형 경량 NPU가 처리하도록 분리했다. Hybe NPU는 주어진 메모리 대역폭을 효과적으로 활용할 수 있는 최소한의 컴퓨팅 유닛만으로 설계되어 Decode 단계에서 하드웨어 효율과 전력 효율을 극대화했다. 이러한 구조적 분리를 통해 GPU는 더 이상 Decode 단계에 필요한 거대한 KV 캐시를 계속 저장할 필요가 없어졌다. 대신 Fine-grained KV Transmission을 통해 생성된 KV 캐시를 즉시 NPU로 전송하여 GPU의 메모리 부담을 크게 줄였다. 또한, 단계 단위 파이프라이닝 기법으로 NPU가 현재 요청의 Decode 단계를 처리하는 동안 GPU는 다음 요청의 Prefill 단계를 미리 처리하여 시스템 전체의 처리량과 가동률을 높였다. 시뮬레이션 결과 Hybe는 NVIDIA H100 GPU 시스템 대비 Phi-3(100K 컨텍스트) 모델에서 최대 2.1배의 속도 향상을, Llama-3(1M 컨텍스트) 모델에서는 최대 3.9배의 에너지 효율 향상을 달성했다.

5. PAPI

PAPI(Parallel Decoding with PIM)는 연산 집약적 또는 메모리 집약적 커널을 적합한 하드웨어 유닛에 동적으로 매핑하는 PIM 기반 이기종 아키텍처이다.

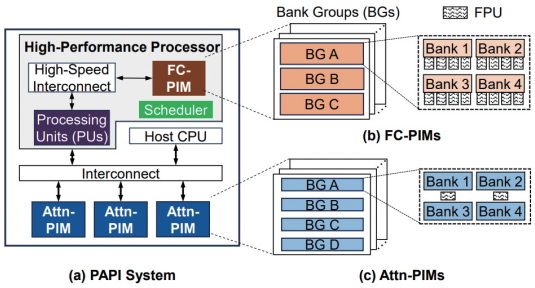
본 연구에서는 LLM 디코딩 커널이 사용자 및 시스템 요구에 따라 연산 집약적 또는 메모리 집약적으로 변할 수 있음을 관찰하고, 이를 기반으로 커널을 특성화하여 런타임 시 가장 적합한 하드웨어 유닛에 동적으로 매핑하였다. 더 나아가, 메모리 집약적 커널 내부에서도 이질성이 크기 때문에 동일한 연산 능력만을 갖춘 PIM 장치로 설계하는 것은 비효율적이며, 서로 다른 연산 능력을 갖춘 하이브리드 PIM 유닛의 필요성을 강조한다.

5.1 Motivation

LLM 추론 작업에서 LLM 디코딩 단계는 실행 시간의 대부분을 차지하며, GPT-3 175B의 모델의 경우 96%에 해당한다. 이 단계의 성능을 향상시키기 위해, 기존 연구들은 주로 Batching과 Speculative Decoding 기법을 활용해 하나의 디코딩 반복에서 여러 토큰을 동시에 생성하는 병렬 디코딩을 구현한다. 동시에 생성되는 디코딩 토큰의 개수는 요청 출력 길이에 따른 메모리 요구량, 서비스 품질과 같은 다양한 사용자 요구사항, Batching 및 Speculative Decoding에 대한 동적 조정이라는 세 가지 주된 요인에 의해 동적으로 변화한다. 이러한 변화는 정적 매핑 방식을 사용하는 이기종 설계를 비효율적으로 만들고, 그 결과 연산 집약적 커널이 PIM 유닛에 잘못 매핑되거나 메모리 집약적 커널이 GPU와 같은 연산 중심 가속기에 잘못 매핑되는 문제가 발생한다.

5.2 Approaches

본 논문은 LLM의 연산 집약적 커널과 메모리 집약적 커널을 모두 효과적으로 처리하기 위해, 호스트 CPU(Central Processing Unit), PIM 메모리 유닛을 갖춘 고성능 프로세서(FC-PIM), 물리적으로 분리된 PIM 유닛(Attn-PIM)으로 구성된 이기종 아키텍처를 제안한다. FC-PIM과 Attn-PIM는 동일한 뱅크 수준 연산 구조와 메모리 계층 구조를 공유하지만, 뱅크



출처 Reprinted from Y. He et al., "Papi: Exploiting dynamic parallelism in large language model decoding with a processing-in-memory-enabled computing system," in Proc. ACM Int. Conf. Archit. Support Program. Lang. Oper. Syst., (Rotterdam, Netherlands), Mar. 2025, pp. 766-782.

그림 2 두 종류의 PIM 가속기를 활용하는 PAPI 시스템의 구조도

당 FPU의 수에서 차이를 두어 서로 다른 연산 특성에 최적화된다. FC-PIM은 연산 집약적 작업을 위해 뱅크당 더 많은 FPU를 배치하고, Attn-PIM은 메모리 집약적 작업을 위해 뱅크당 더 적은 FPU를 배치한다(그림 2).

이러한 구조적 차이는 두 장치의 역할 분담으로 이어진다. FC-PIM은 높은 연산 성능을 요구하는 FC 커널을, Attn-PIM은 대규모 메모리 용량을 필요로 하는 어텐션 커널을 효과적으로 처리하며, 두 장치는 서로 다른 병렬화 수준을 담당하면서 상호 보완적으로 작동한다. 커널 매핑은 이러한 특성을 기반으로 이루어진다. 어텐션 커널은 본질적으로 메모리 집약적이므로 Attn-PIM에 고정적으로 매핑되고, FC 커널은 상황에 따라 연산 집약적일 수도 메모리 집약적일 수도 있으므로, 스케줄러가 병렬화 수준을 모니터링해 저비용 식별 단계를 수행한 뒤, Processing Unit 또는 FC-PIM에 동적으로 매핑된다.

세 가지 널리 사용되는 LLM(LLaMa-65B, GPT-3 66B, GPT-3 175B)을 대상으로 한 실험 결과, PAPI는 최신 GPU+PIM 기반 LLM 가속기에 비해 1.8배, 최신 PIM 전용 LLM 가속기에 비해 11.1배의 속도 향상을 달성하였다.

IV. 결론

본고에서는 PIM을 이용한 효율적인 LLM 추론 연구 동향을 살펴보았다. 메모리 병목인 LLM 추론이 최근 폭발적으로 그 수요가 증가함에 따라, 데이터 이동을 최소화하여 지연시간과 전력소비를 줄이는 PIM 가속기를 LLM 추론의 셀프 어텐션과 FFN 블록에서 서로 다른 GEMV 연산에서 활용하려는 노력들이 최근 활발히 이루어졌음을 살펴보았다. 전통적인 DRAM에서 활용될 때, 새롭게 HBM에서 활용될 때 PIM의 하드웨어적 설계와 시스템적 고려사항을 살펴보았고, GPU 또는 NPU와 효과적으로 동시 수행될 수 있는 분할 및 스케줄링 기법들, 특히 GPU 또는 NPU의 메모리로서 접근과 PIM으로서의 연산이 충돌하지 않기 위한 스케줄링 기법들을 살펴보았으며, 단일 종류가 아니라 PIM 가속기도 메모리 병목의 정도에 따라 이기종으로 존재할 경우의 효용성을 주장하는 최신 연구도 살펴보았다.

PIM이 성공적으로 확산될 수 있을지는 단순히 PIM 하드웨어적인 설계와 구현을 넘어서, 기존 GPU나 NPU에 PIM이 추가되는 이기종 시스템에서 어떻게 LLM 추론 레이어 블록들을 효과적으로 분할하고 스케줄링할 수 있을지, 이러한 설정이 다양한 모델과 시스템이 주어질 때 개발자와 연구자를 비롯한 시스템 사용자가 얼마나 손쉽게 그 최적 설정을 도출하여 효과적으로 가속할 수 있을지에 달려있을 것이다.

용어해설

연산 집약도 메모리 접근량 대비 연산량의 비율

통합 메모리 xPU와 PIM 등의 이기종 프로세서가 동일한 주소 공간을 공유하는 메모리 시스템

분할 메모리 각 프로세서가 독립된 메모리를 가지고 있으며, 다른 프로세서의 메모리에 접근하려면 통신이 필요한 메모리 시스템

Prefill 대규모 언어 모델 추론에서 입력 프롬프트 전체를 한 번에 처리하는 초기 단계

Decode 대규모 언어 모델 추론에서 Prefill 이후 토큰을 한 개씩 순차적으로 생성하는 단계

참고문헌

[1] T. Dao et al., "Flashattention: Fast and memory-efficient exact attention with io-awareness," in Proc. 36th Int. Conf. Neural Inf. Process., (New Orleans, LA, USA), Nov, 2022, pp. 16344-16359.

[2] Y. Leviathan et al., "Fast inference from transformers via speculative decoding," in Proc. Int. Conf. Mach. Learn., (Honolulu, HI, USA), Jul. 2023, pp. 19274-19286.

[3] V. Seshadri et al., "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," in Proc. Annu. IEEE/ACM Int. Symp. Microarchitecture, (Cambridge, MA, USA), Oct. 2017, pp. 273-287.

[4] A.M. Bidgoli et al., "NeuroPIM: Felxible Neural Accelerator for Processing-in-Memory Architectures," in Proc. Int. Symp. Design Diagn. Electron. Circuits Syst., (Tallinn, Estonia), May, 2023, pp. 51-56.

[5] Y. Kwon et al., "System architecture and software stack for GDDR6-AiM," in Proc. IEEE Hot Chips 34 Symp., (Cupertino, CA, USA), Aug. 2022, pp. 1-25.

[6] S.H. Lee et al., "Hardware architecture and software stack for PIM based on commercial DRAM technology: Industrial product," in Proc. Annu. Int. Symp. Comput. Archit., (Valencia, Spain), Jun. 2021, pp. 43-56.

[7] J.G.-Luna et al., "Benchmarking a new paradigm: An experimental analysis of a real processing-in-memory architecture," arXiv preprint, 2021. doi: 10.48550/arXiv.2105.03814

[8] G. Heo et al., "Neupims: Npu-pim heterogeneous acceleration for batched llm inferencing," in Proc. ACM Int. Conf. Archit. Support Program. Lang. Oper. Syst., (La Jolla, CA, USA), Apr. 2024, pp. 722-737.

[9] M.S. Seo et al., "Ianus: Integrated accelerator based on NPU-PIM unified memory system," in Proc. ACM Int. Conf. Archit. Support Program. Lang. Oper. Syst., (La Jolla, CA, USA), Apr. 2024, pp. 545-560.

[10] J.H. Park et al., "Attacc! unleashing the power of pim for batched transformer-based generative model inference," in Proc. ACM Int. Conf. Archit. Support Program. Lang. Oper. Syst., (La Jolla, CA, USA), Apr. 2024, pp. 103-119.

[11] S.J. Moon et al., "Hybe: GPU-NPU Hybrid System for Efficient LLM Inference with Million-Token Context Window," in Proc. Annu. Int. Symp. Comput. Archit., (Tokyo, Japan), Jun. 2024, pp. 808-820.

[12] Y. He et al., "Papi: Exploiting dynamic parallelism in large language model decoding with a processing-in-memory-enabled computing system," in Proc. ACM Int. Conf. Archit. Support Program. Lang. Oper. Syst., (Rotterdam, Netherlands), Mar. 2025, pp. 766-782.

[13] Y.G. Kim et al., "Ramulator: A fast and extensible DRAM simulator," IEEE Comput. Archit. Lett., vol. 15, no. 1, 2015, pp. 45-49.

[14] H. Luo et al., "Ramulator 2.0: A modern, modular, and extensible dram simulator," IEEE Comput. Archit. Lett., vol. 23, no. 1, 2023, pp. 112-116.