칩렛 이종집적 첨단패키지 기반 LLM 가속기 설계 동향

Technological Trends in Large Language Model Accelerator Design based on Chiplet Heterogeneous Integration and Advanced Packaging

장명재 (M.J. Jang, myeongjae0409@etri.re.kr) PIM인공지능반도체연구실 선임연구원
권현정 (H.J. Kwon, kwonhj@etri.re.kr) PIM인공지능반도체연구실 선임연구원
최재웅 (J.W. Choi, cjw921113@etri.re.kr) PIM인공지능반도체연구실 연구원
천민규 (M.G. Cheon, mingyu@etri.re.kr) PIM인공지능반도체연구실 석사후연수연구원
한진호 (J.H. Han, soc@etri.re.kr) PIM인공지능반도체연구실 책임연구원/실장

ABSTRACT

Large Language Models (LLMs) have become a key application in AI. Global IT companies, including Meta, Apple, and Google, have joined the AI revolution by developing their own LLMs and related applications. The use of LLMs for on-device AI in mobile and edge devices has led to new era of practical AI environments. Despite these advancements of LLM applications, the significant computational demands of LLMs continue to pose challenges for LLM inference systems and their accelerators. To address these challenges, LLM accelerators and architectural designs have been proposed to support LLM-based applications with rapid response times. In this paper, we observe recent technological trends in LLM accelerator designs, including distributed computing for efficient data flow, mixed-precision general matrix multiplication using low-bit weights, and the use of a MX Format combined with advanced quantization techniques. LLM accelerator designs are becoming various and complex as they aim to achieve performance, power consumption, and thermal and carbon emission constraints. Understanding these latest design trends is crucial for the development of efficient LLM accelerators and their inference systems.

KEYWORDS AI, LLM, MX Format, 가속기 설계, 데이터 압축, 모델 압축, 분산 처리, 양자화

I. 서론

LLM(Large Language Model)은 현재 실생활에서 다양한 어플리케이션 서비스로 활용되고 있는 핵심 인공지능 기술이 되었다. Meta, Apple, Google 등 많은 글로벌 IT 기업들에서 자신들이 직접 설계하고 학습한 다양한 형태의 LLM과 이를 바탕으로 한 서비스를 제공하고 있다. 제공하는 서비스의 형태는 데이터 센터와 서버급 연산 자원을 활용한 하이퍼스케일 AI(Hyperscale AI)부터 모바일 기기나 엣지 디

* DOI: https://doi.org/10.22648/ETRI.2025.J.400502
* 이 연구는 2025년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원(IITP)의 지원을 받아 수행된 연구임[RS-2022-II221036. 페타플롭스급 성능 기가바이트급 메모리 융합 초고성능 PIM 프로세서 반도체 개발].

바이스(Edge Device)상에서 사용자의 입력을 받아 처리하는 온디바이스 AI(On-Device AI)까지 그 크기와 형태 및 활용 목적이 다양하다.

이러한 시대 상황은 효율적이고 빠른 LLM 기반 서비스의 제공에 대한 요구를 만들어내고 있다. 하지만 LLM의 발전 과정에서 LLM의 크기와 한 번에 처리해야 하는 데이터의 양이 계속 증가하고 있다. 올해 3월에 공개되어 세계적인 돌풍을 일으켰던 DeepSeek-V3의 파라미터 수는 685B(Billion, 10억)이며[1], Moonshot AI에서 발표한 Kimi K2는 전체 파라미터 수가 1T(Trillion, 1조)를 넘어섰다[2]. 상용 컴퓨터로는 LLM의 학습은 물론 추론도 하기 어려운 수준에 도달하였으며, 기업들도 막대한 자금을 투자하여 서버급 GPU를 다수 사용하여야만 요구사항에 부합하는 처리 속도를 달성할 수 있게 되었다. 이 과정에서 비용, 발열, 전력, 탄소 배출 등 부차적인 문제도 함께 발생하고 있다.

이러한 LLM 추론 시스템의 효율성 증대를 위해서는 적절한 LLM 가속기의 설계와 활용이 필수적이다. 인공지능 가속기는 예전부터 많이 연구되었고, Google의 TPU[3]나 삼성전자의 Exynos NPU(Neural Processing Unit)[4] 등 이미 상용화된 인공지능 가속기도 존재한다. 하지만 계속해서 발전하고 크기가 커지고 있는 LLM을 효율적으로 처리하기 위해서는 이에 맞게 진보된 LLM 가속기가 필요하다.

본고에서는 효율적인 LLM 가속기를 위한 최신 설계 동향을 분석한다. 나아가 현재 ETRI 내에서 수행되고 있는 LLM 가속기 설계 과제의 방향성과 앞으로의 계획을 살펴본다.

II. LLM 가속기 설계 동향

효율적인 LLM 가속기 설계 기법에는 다양한 형태가 존재한다. II장에서는 최신 LLM 가속기 설계 기법으로써 (1) 양자화(Quantization), (2) 데이터 재사용을 고려한 분산 처리(Distributed Computing), (3) 극단적 저정밀도 자료형 기반의 mpGEMM(mixed-precision General Matrix Multiplication) 연산, 그리고 (4) LLM 데이터 병목 현상 해결을 위한 데이터 압축을 분석한다.

1. LLM 데이터 양자화 기법

LLM 데이터 양자화는 LLM 가속기 설계에서 자주 사용하는 인공지능 연산 최적화 기법이다. 양자화는 크게 자료형(Datatype)과 양자화 방식의 두 가지를 고려하여 수행한다.

먼저, 자료형은 연산 성능 및 효율성 증대를 위해 연산 오버헤드가 큰 32-/64-Bit 부동소수점 자료형(FP32/64) 대신 비트 수가 적고 연산기가 단순한 정수 자료형(INT1/2/4)으로 데이터를 변환한다.

자료형의 정밀도 하락으로 인하여 LLM의 정확도(Accuracy)가 낮아지는 문제가 발생한다. 이를 적절한 양자화 방식을 적용하여 해결한다. 양자화 방식은 크게 PTQ(Post-Training Quantization)[5]와 QAT(Quantization-Aware Training)[6]로 구분된다. PTQ는 학습이 종료된 LLM에 대하여 추가적인 재학습 없이 양자화만 적용하여 바로 추론하는 방식이다. 재학습이 없어 오버헤드가 적고 양자화가 비교적 간단하여 LLM 가속기 설계에서 주로 고려된다. QAT는 양자화 결과로 낮아지는 LLM 정확도를 추가적인 재학습으로 복원하는 과정을 거친다. 재학습을 수행하기 때문에 PTQ 대비 오버헤드가 크지만, 양자화 이후에 LLM의 크기나 정확도 측면에서 더 좋은 결과를 보인다. 재학습을 통해 양자화에 최적화된 LLM 및 전용 LLM 가속기를 설계할 수 있다. 자료형의 비트 수가 적을수록 LLM 정확도에 영향이

표 1 최신 LLM 가속기 설계 연구 내 양자화 기법

가속기 설계 기법	양자화 자료형	양자화 방식
Oaken[9]	INT4/5 activation	PTQ + 그룹 단위 양자화
MicroScopiQ[12]	MXINT2/4 weight + MXINT2/4/8 activation	PTQ + Inlier-and-Outlier 양자화
Transitive Array[16]	INT4/8 weight + INT8 activation	PTQ
LUT Tensor Core[17]	INT1/2/4 weight + FP8/16 activation	QAT[1)]
Ecco[22]	–[2)]	QAT + *k*-Means 패턴 양자화

1) 제안 기법이 양자화를 포함하고 있지는 않으나, QAT가 적용된 LLM을 대상으로 하고 있음
2) 양자화와 데이터 압축을 동시 수행한 기법으로 특정 자료형에 국한되지 않음

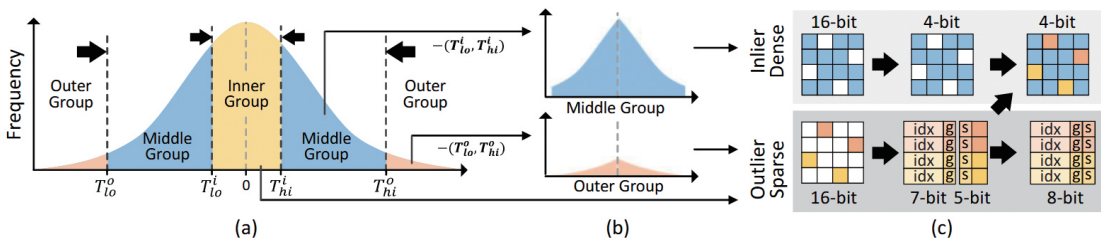
크기 때문에 극단적 저정밀도 자료형을 이용한 양자화에서는 QAT가 더 선호된다.

표 1은 본고에서 분석하는 최신 LLM 가속기 설계 연구에서 사용한 양자화 기법을 정리한 것이다. 최신 LLM 가속기 설계 연구는 양자화를 제안 기법에 포함하거나 양자화된 LLM을 대상으로 진행하는 경우가 많다. 본고에서 양자화 기법으로 분류하지 않은 연구도 양자화에 대한 부분이 포함되어 있다.

1.1 LLM 데이터 분포를 고려한 그룹 단위 양자화

Transformer[7]는 최신 LLM의 가장 기본이 되는 단위 구조이다. Transformer의 내부에는 각 단어 토큰(Word Token)의 중요도를 계산하는 어텐션 메커니즘(Attention Mechanism)이 존재한다. 어텐션 메커니즘에서는 쿼리(Query), 키(Key), 밸류(Value)로 구분된 데이터에 대한 GEMM 연산이 이루어지며, 이 중 키와 밸류의 연산 결과는 여러 Transformer를 거치며 반복 사용된다. 반복적인 재연산과 불필요한 메모리 접근을 최소화하기 위하여 키, 밸류의 연산 결과를 캐싱하는데, 이를 KV 캐시(KV Cache)라고 한다. KV 캐시는 최신 LLM 가속기 설계에서 최적화의 대상으로 자주 고려된다[8].

Oaken[9]은 KV 캐시를 양자화하여 불필요한 메모리 접근을 줄이고 전반적인 LLM 추론 속도를 향상하는 기법이다. KV 캐시를 양자화하는 과정에서 발생하는 LLM 정확도 하락을 방지하기 위해 그림 1과 같이 KV 캐시의 값을 세 그룹으로 나누어 개별적으로 양자화를 수행한다. LLM 정확도에 많은 영향을 주는 Outlier 값을 고려하여 양자화하는 기법



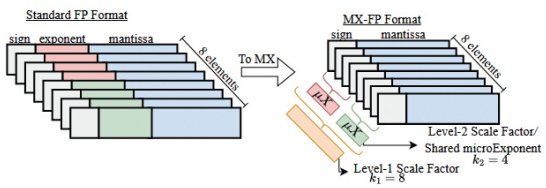
출처 Reprinted from M.S. Kim et al., "Oaken: Fast and Efficient LLM Serving with Online-Offline Hybrid KV Cache Quantization," in Proc. Int. Symp. Comput. Archit., (Tokyo, Japan), Jun. 2025, pp. 482–497.

그림 1 Oaken[9]의 KV 캐시에 대한 그룹 단위 양자화: (a) 한곗값(Threshold)을 고려한 온라인-오프라인 복합 양자화 (b) 그룹 시프트(Group-Shift) 양자화 (c) Dense-and-Sparse 인코딩

들이 다수 제안되었으나[10-12], 이 기법은 세 개 이상의 그룹으로 데이터를 나누어 양자화를 수행하였다는 점에서 차이가 있다. LLM 정확도에 영향이 적으면서 비율이 큰 중앙 그룹(Middle Group)은 INT4 자료형으로 양자화하고, 나머지 내부/외부 그룹(Inner/Outer Group)은 INT5 자료형으로 양자화를 수행하여 정밀도를 보존한다(그림 1(a)). 각 그룹은 효율적인 양자화를 위하여 0 값을 기준으로 그룹 시프트(Group-Shift)를 수행한다(그림 1(b)). 연속된 데이터를 여러 그룹으로 나누어 양자화를 수행함으로써 발생하는 메모리 관리 및 접근 시 오버헤드를 최소화하기 위하여 Dense-and-Sparse 인코딩을 추가로 적용한다(그림 1(c)).

1.2 MX format 기반 LLM 양자화

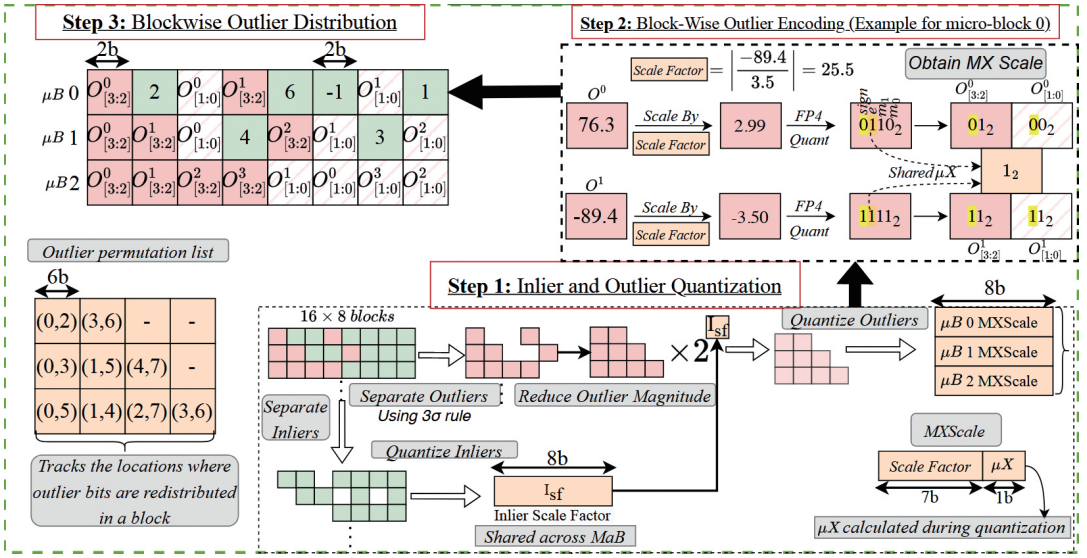
MX(Microscaling) Format[12,13]은 효율적인 인공지능 프로세스 연산을 위해 Microsoft에서 제안한 새로운 자료형이다. 그림 2는 연속된 8개의 FP



출처 Reprinted from A. Ramachandran et al., "Microscopiq: Accelerating foundational models through outlier-aware microscaling quantization," in Proc. Int. Symp. Comput. Archit., (Tokyo, Japan), Jun. 2025, pp. 1193-1209.

그림 2 단일 블록 MX(Microscaling) Format

Format 데이터를 한 개의 블록(Block)으로 보고 MX Format을 적용한 예시이다. 단일 블록 내의 데이터는 한 개의 공유값(Shared Scale Factor)을 가지며, 각 데이터는 공유값과 기존값의 차이만을 저장하는 형태로 변환된다. 데이터 전체에 영향을 주는 공유값은 고정밀도로 표현하고, 차이값은 저정밀도로 표현하여 전체적인 데이터의 정밀도는 유지하면서도 각 데이터를 저장하는 비트 수는 효율적으로 줄일 수 있다. 데이터 표현의 단순화를 위하여, 공유



출처 Reproduced from A. Ramachandran et al., "Microscopiq: Accelerating foundational models through outlier-aware microscaling quantization," in Proc. Int. Symp. Comput. Archit., (Tokyo, Japan), Jun. 2025, pp. 1193-1209.

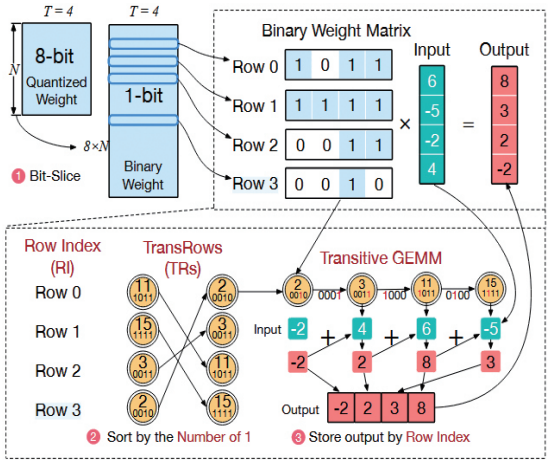
그림 3 MicroScopiQ[12]의 MX format 기반 Inlier-and-Outlier 양자화의 적용 예시

값이 FP Format상의 Exponent를 대체하며 차이값은 Mantissa로만 구성할 수 있다. 또한, 공유값을 여러 단계로 생성하여 블록의 크기를 조정하거나 여러 블록 간에도 공유값을 생성할 수도 있다. 연속된 다수의 데이터를 한 번에 처리하기 때문에 텐서(Tensor)나 벡터(Vector) 단위 병렬처리에 효과적이다.

이와 같은 장점을 가지고 있는 MX Format을 LLM 가속기에도 활용하고자 하는 연구들이 다수 발표되었다[12,14,15]. 이 중, MicroScopiQ[12]는 LLM Weight와 Activation 데이터 모두 MX Format을 효율적으로 적용하는 양자화 방법과 이를 바탕으로 구현한 LLM 가속기 아키텍처를 포함한다. 그림 3은 LLM Weight 행렬에 MicroScopiQ의 양자화를 적용하는 예시이다. 이 기법은 Outlier를 고려한 양자화를 사용한다. 일정 기준 이상의 값을 Outlier(적색)로, 나머지 값을 Inlier(녹색)로 한다. LLM 정확도에 영향이 적은 Inlier는 MXINT2(차이값에 대한 자료형이 INT2인 MX Format)를 적용한다(Step 1). 공유값은 8Bits를 사용한다. Outlier는 높은 정밀도를 위해 MXINT4를 적용한다(Step 2). Outlier의 차이값에 대한 비트 수가 Inlier의 두 배이므로, 이를 함께 저장하기 위해 정확도에 영향이 적은 Inlier를 프루닝(Pruning)하고 해당 공간에 Outlier의 비트를 분할하여 저장한다(Step 3). 이전에 분석하였던 Oaken[9]과 유사하게 LLM 데이터를 두 그룹으로 나누어 서로 다른 정밀도의 자료형으로 양자화하고 메모리 효율성을 위해서 복합적으로 저장하는 방식이다.

2. LLM 데이터 재사용을 고려한 연산기 분산 처리

LLM을 포함한 모든 인공지능 가속기는 빠른 처리와 높은 병렬성을 위하여 복수의 연산기를 사용한다. 이 과정에서 효율적인 병렬처리를 위한 데이터 전송 및 프로세스 스케줄링(Scheduling)이 가속기 설계 수준에서 필수적으로 고려되어야 한다.



출처 Reprinted from C. Guo et al., "Transitive Array: An Efficient GEMM Accelerator with Result Reuse," in Proc. Int. Symp. Comput. Archit., (Tokyo, Japan), Jun. 2025, pp. 990-1004.

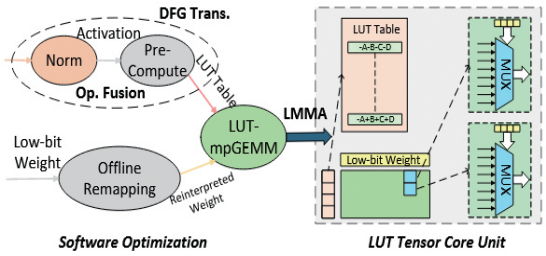
그림 4 GEMM 연산 과정에서 발생하는 연산 결과의 중복과 Transitive Array[16]를 이용한 중복된 값의 전달 및 연산 최적화

반복적인 GEMM 연산으로 구현되는 LLM의 특성상, 데이터 재사용(Data Reuse)이 빈번하게 발생한다. 불필요한 메모리 접근을 줄이기 위해서는 재사용이 가능한 데이터를 복수의 연산기에서 최대한 활용할 수 있도록 가속기를 설계할 필요가 있다. Transitive Array[16]는 GEMM 연산 과정에서 발생하는 데이터 재사용을 극대화하는 LLM 가속기 구조이다. 그림 4와 같이 해당 기법에서는 INT8 Weight를 0, 1로만 구성되는 비트 단위로 분할한다. 각 비트에 대하여 연산의 결과가 중복되어 재사용이 가능한 경우를 그래프 형태로 계산하고 최적화된 경로를 따라 현재의 연산 결과를 다음 연산기에 전달하면서 연산을 수행한다. 데이터 재사용으로 인하여 반복적인 메모리 접근을 최소화할 뿐만 아니라 이전에 연산이 종료된 결과를 활용함으로써 연산 횟수도 감소한다.

3. 극단적 저정밀도 자료형 기반 mpGEMM 가속기

Ⅱ장 1절에서 다루었던 LLM 데이터 양자화를 INT1과 같은 극단적인 저정밀도 자료형으로 적용하면 LLM 가속기를 설계하는 과정에서 많은 이득을 얻을 수 있다. 단순히 하드웨어 오버헤드가 적은 연산기를 사용하는 수준에서 나아가, 연산 자체를 수행하지 않고 Look-Up Table(LUT) 접근으로 처리하는 것이 가능해진다[17-19]. 극단적인 저정밀도 자료형은 비트 수가 적어 표현 가능한 값의 종류가 제한적이다. 따라서 모든 표현 가능한 값에 대하여 미리 연산을 수행한 뒤, 이 결과를 LUT에 저장해두고 동일 연산이 발생하면 LUT Look-Up을 수행하여 연산을 대신한다.

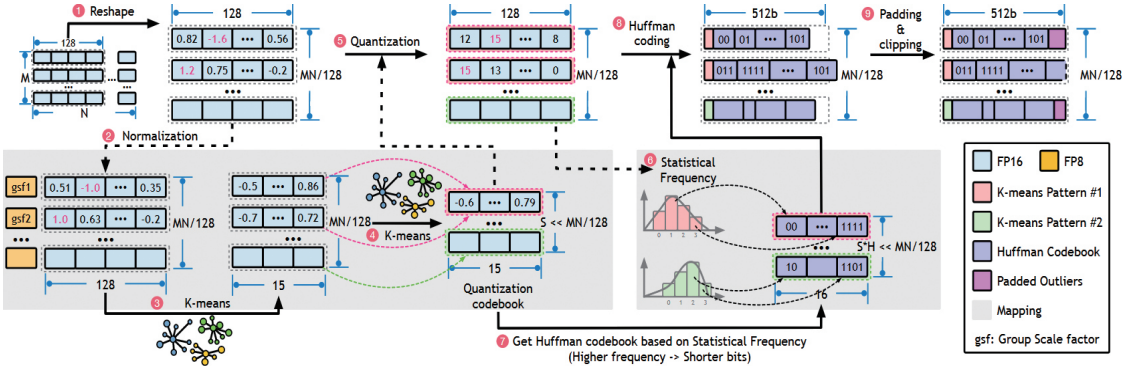
그림 5의 LUT Tensor Core[17]는 LUT 기반 mpGEMM 연산을 소프트웨어와 하드웨어 수준에서 통합 최적화한 LLM 가속기 설계 기법이다. LUT 기반 mpGEMM 연산에서 가장 큰 오버헤드를 가지는 부분은 LUT를 채우기 위해 가능한 모든 경우의 수를 계산하는 과정이다. 이 과정을 이전 레이어(Layer) 연산과 함께 한 번에 수행하도록 하여 오버헤드를 감소시킨다.



출처 Reprinted from Z. Mo et al., "LUT Tensor Core: A Software-Hardware Co-Design for LUT-Based Low-Bit LLM Inference," in Proc. Int. Symp. Comput. Archit., (Tokyo, Japan), Jun. 2025, pp. 514-528.

그림 5 LUT Tensor Core[17]의 소프트웨어-하드웨어 통합 최적화

LUT를 저장하는 메모리 공간 오버헤드도 LUT 기반 mpGEMM 연산에서 해결해야 할 주된 문제이다. 해당 기법은 극단적 저정밀도 자료형의 Weight 벡터를 다시 리맵핑(Remapping)하여 0 값을 기준으로 대칭이 되도록 만들고, LUT는 양수만 저장하고 음수는 부정 연산(NOT Gate)으로 처리하여 LUT 크기를 절반으로 줄인다. 마지막으로 LUT Look-Up을 통한 연산을 수행하는 멀티플렉서(MUX: Multiplexer)를 병렬로 설계하여 빠른 처리 속도와 LUT 재사용률을 극대화한다.



출처 Reprinted from F. Cheng et al., "Ecco: Improving Memory Bandwidth and Capacity for LLMs via Entropy-Aware Cache Compression," in Proc. Int. Symp. Comput. Archit., (Tokyo, Japan), Jun. 2025, pp. 793-807.

그림 6 Ecco[22]의 *k*-Means 패턴을 활용한 양자화와 Huffman 인코딩 압축 기법

4. 메모리 병목 현상 해결을 위한 LLM 데이터 압축

LLM에 대한 압축은 크게 모델 압축과 데이터 압축으로 나눌 수 있다. 모델 압축은 LLM 자체를 압축하는 방식으로써 프루닝이나 양자화가 대표적이다. LLM 구조가 변화하여 정확도에 직접적인 영향을 줄 수 있다. 반대로 데이터 압축은 LLM 내의 다양한 데이터를 비트 수준에서 압축하는 방식이다. 전통적인 컴퓨터 아키텍처에서부터 메모리 효율성을 위해 BPE(Byte Pair Encoding)[20], LZ4(Lempel-Ziv 4)[21] 등의 데이터 압축 기법들이 사용되었다.

Ecco[22]는 LLM 양자화와 함께 Huffman 인코딩[23]을 함께 적용하여 LLM 데이터로 인한 메모리 공간 및 대역폭 문제를 해결하는 설계 기법이다. 그림 6은 해당 기법의 양자화 및 압축 과정을 나타낸다. ❶ 입력된 LLM 데이터를 128개의 그룹으로 나누고, ❷ 양자화를 수행하여도 값의 변화가 크지 않도록 정규화(Normalization)를 수행한다. 이 과정에서 그룹별로 스케일 팩터(Scale Factor)를 추출하여 값의 범위를 조정한다. ❸ 그룹 내 값들과 ❹ 그룹 간에 *k*-Means 패턴 클러스터링을 적용하고, ❺ 이 패턴을 이용하여 양자화를 수행한다. ❻ 양자화된 값의 분포를 바탕으로 ❼ *k*-Means 패턴에 Huffman 인코딩을 수행하고, ❽ 생성된 Huffman 코드를 양자화된 값에 적용한다. ❾ 마지막으로 메모리 접근 크기에 맞추어 패딩(Padding)이나 클리핑(Clipping)을 수행하여 각 그룹의 크기를 동일하게 맞춘다.

III. 칩렛 이종집적 첨단패키지 기반 LLM 가속기 개발

Ⅱ장에서 다루었던 최신 LLM 가속기 설계 기법들은 양자화를 포함하거나 양자화된 LLM을 상정하고 있다. 이미 NVIDIA의 블랙웰(Blackwell) 아키텍처의 텐서 코어(Tensor Core)는 FP4/6 등의 저정밀도 연산을 지원하고 있으며, MX Format 역시 지원할 예정이다[24]. ETRI 내에서도 양자화를 기반으로 하는 LLM 가속기의 설계 및 구현 과제를 준비하고 있다. 이미 FP32와 BF16 자료형을 복합적으로 연산할 수 있는 칩렛(Chiplet) 이종집적 첨단패키지 기반 LLM 가속기를 개발하고 있으며, 이를 바탕으로 비트 수가 더 적은 저정밀도 자료형을 추가로 지원하는 것을 목표로 하고 있다.

최근의 LLM 가속기는 크게 하이퍼스케일 AI(Hyperscale AI)를 위한 서버급 가속기와 온디바이스 AI(On-Device AI)를 위한 경량 가속기로 구분된다. 서버급 가속기는 병렬성의 극대화를 위해 MX Format과 같은 자료형을 적극적으로 활용하며, 초거대 LLM의 정확도를 고려하여 Outlier 기반의 양자화와 이를 연산하기 위한 아키텍처를 가지고 있다[9,12]. 반대로 경량 가속기는 극단적 저정밀도 자료형으로 양자화된 LLM을 대상으로 하고 있으며, 전력 소모 최소화를 위해 GEMM 연산을 LUT Look-Up으로 대체하는 등의 모습을 보인다[16,17]. ETRI에서는 두 가지 LLM 가속기의 방향성에 대하여 심도 있는 분석을 진행하고 있으며, 지속적인 연구 개발 가능성과 사회의 요구에 맞추어 차세대 LLM 가속기 설계 및 개발을 계획 중이다. 서버급 가속기는 MX Format 및 Outlier 양자화를 효과적으로 지원하면서 최근의 초거대 LLM이 가지고 있는 Mixture-of-Expert[1], Chain-of-Thought[25] 등의 특성을 효과적으로 연산할 수 있도록 연산 스케줄링 및 분산 처리 기능을 추가할 계획이다. 또한, 메모리 저장 공간 및 대역폭 최적화를 위하여 LLM 데이터를 효율적으로 압축하는 방향도 고려 중이다. 경량 가속기의 경우, LUT 기반의 mpGEMM 연산을 지원하고 런타임(Runtime) 양자화 및 복원이 가능

하도록 관련 모듈을 탑재할 예정이다. Microsoft의 BitNet[26]과 같은 극단적 저정밀도 자료형으로 양자화된 LLM은 정확도의 하락을 방지하기 위하여 특별하게 설계된 레이어를 사용하고 있다. 이와 같은 특수 레이어를 효과적으로 처리할 수 있는 아키텍처 역시 논의되고 있다.

IV. 결론

계속된 LLM의 발전과 이를 활용한 실제 서비스 환경의 폭발적인 증가는 다양한 형태의 시스템 요구사항을 만들어내고 있다. 이러한 다양한 문제들을 인공지능 가속기 기술 개발 초기에 대두되었던 GEMM 연산 기반의 다중 병렬 연산기 구조만으로 해결하는 것은 한계가 있다. 본고에서 분석한 효율적인 LLM 양자화 및 저정밀도 자료형 연산기 구조를 활용하고, 이를 기반으로 효과적인 분산 처리가 가능한 LLM 가속기를 설계 및 개발하는 것이 필요하다. 최신 LLM 가속기 설계 동향을 이해하고 실제 가속기 개발에 활용하여야만 다가올 인공지능 기반 사회에서 빠른 인공지능 서비스와 효율적인 연산 시스템 운용이 가능할 것이다.

용어해설

mpGEMM(mixed-precision GEMM) 정밀도가 다른 두 입력 행렬의 곱셈 연산으로, 최근 저정밀도 Weight와 고정밀도 Activation으로의 LLM 양자화가 대두되면서 LLM 가속기가 핵심적으로 수행하여야 할 연산이 됨

MX(Microscaling) Format Microsoft에서 제안한 새로운 자료형. 여러 개의 데이터를 한 개의 공유값과 각 데이터의 차이로 나타냄. 연속된 데이터를 블록 형태로 처리하여 텐서 및 벡터 기반 병렬처리에 효과적이며, 추후 NVIDIA GPU에서 MX Format을 지원할 것으로 알려져 있음

LUT(Look-Up Table) 추가적인 값의 업데이트 없이 단순히 저장된 값의 접근 및 확인만을 위한 테이블 형태의 자료 구조. 실제 연산기를 이용하여 연산을 수행하지 않고 LUT 접근으로 연산을 대체하여 LLM 가속기 구조의 단순화 및 추론 성능 최적화가 가능함

참고문헌

[1] D. Guo et al., "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," arXiv preprint, 2025. doi: 10.48550/arXiv.2501.12948

[2] E. Gibney, "'Another DeepSeek moment': Chinese AI model Kimi K2 stirs excitement," Nature, vol. 643, 2025, pp. 889-890. https://www.nature.com/articles/d41586-025-02275-6

[3] N. Jouppi et al., "TPU v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings," in Proc. Int. Symp. Comput. Archit., (Orlando, FL, USA), Jun. 2023, pp. 1-14.

[4] J.O. Song et al., "7.1 An 11.5 TOPS/W 1024-MAC butterfly structure dual-core sparsity-aware neural processing unit in 8nm flagship mobile SoC," in Proc. IEEE Int. Solid-State Circuits Conf., (San Francisco, CA, USA), Feb. 2019, pp. 130-132.

[5] Y. Nahshan et al., "Loss aware post-training quantization," Mach. Learn., vol. 110, no. 11-12, 2021, pp. 3245-3262.

[6] M. Nagel et al., "Overcoming oscillations in quantization-aware training," in Proc. Int. Conf. Mach. Learn., (Baltimore, MD, USA), Jul. 2022, pp. 16318-16330.

[7] A. Vaswani et al., "Attention is all you need," in Proc. Adv. Neural Inf. Process. Syst., (Long Beach, CA, USA), vol. 30, Dec. 2017.

[8] L. Shi et al., "Keep the cost down: A review on methods to optimize LLM's KV-cache consumption," arXiv preprint, 2024. doi: 10.48550/arXiv.2407.18003

[9] M.S. Kim et al., "Oaken: Fast and Efficient LLM Serving with Online-Offline Hybrid KV Cache Quantization," in Proc. Int. Symp. Comput. Archit., (Tokyo, Japan), Jun. 2025, pp. 482-497.

[10] C.H. Lee et al., "OWQ: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models," in Proc. AAAI Conf. Artif. Intell., (Vancouver, BC, Canada), Vol. 38, No. 12, Mar. 2024, pp. 13355-13364.

[11] C. Guo et al., "OliVe: Accelerating large language models via hardware-friendly outlier-victim pair quantization," in Proc. Int. Symp. Comput. Archit., (Orlando, FL, USA), Jun. 2023, pp. 1-15.

[12] A. Ramachandran et al., "Microscopiq: Accelerating foundational models through outlier-aware microscaling quantization," in Proc. Int. Symp. Comput. Archit., (Tokyo, Japan), Jun. 2025, pp. 1193-1209.

[13] B.D. Rouhani et al., "Microscaling data formats for deep learning," arXiv preprint, 2023. doi: 10.48550/arXiv.2310.10537

[14] D. Gorodecky and L. Sousa, "Hardware for converting floating-point to the microscaling (MX) format," arXiv preprint, 2024. doi: 10.48550/arXiv.2411.03149

[15] E. Samson et al., "Exploring FPGA designs for MX and beyond," in Proc. Int. Conf. Field-Program. Logic Appl., (Torino, Italy), Sep. 2024, pp. 304-310.

[16] C. Guo et al., "Transitive Array: An Efficient GEMM Accelerator with Result Reuse," in Proc. Int. Symp. Comput. Archit., (Tokyo, Japan), Jun. 2025, pp. 990-1004.

[17] Z. Mo et al., "LUT Tensor Core: A Software-Hardware Co-Design for LUT-Based Low-Bit LLM Inference," in Proc. Int. Symp. Comput. Archit., (Tokyo, Japan), Jun. 2025, pp. 514-528.

[18] G.H. Park et al., "Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models," arXiv preprint, 2022. doi: 10.48550/arXiv.2206.09557

[19] S.Y. Um et al., "Dial: An Energy-Efficient Dram in-Memory Computing Accelerator with Compact Partial Product Lut and Twisted Differential Adc," in Proc. Symp. VLSI Technol. Circuits, (Kyoto, Japan), Jun. 2025, pp. 1-3.

[20] K. Bostrom and G. Durrett, "Byte pair encoding is suboptimal for language model pretraining," arXiv preprint, 2020. doi: 10.48550/arXiv.2004.03720

[21] J.H. Kim and J.D. Cho, "Hardware-accelerated fast lossless compression based on LZ4 algorithm," in Proc. Int. Conf. Digit. Signal Process., (Jeju Island, Republic of Korea), Feb. 2019, pp. 65-68.

[22] F. Cheng et al., "Ecco: Improving Memory Bandwidth and Capacity for LLMs via Entropy-Aware Cache Compression," in Proc. Int. Symp. Comput. Archit., (Tokyo, Japan), Jun. 2025, pp. 793-807.

[23] D.A. Huffman, "A method for the construction of minimum-redundancy codes," Resonance, Vol. 11, 2006, pp. 91-99.

[24] NVIDIA Blackwell Tensor Cores Fifth Generation. https://www.nvidia.com/en-us/data-center/tensor-cores/

[25] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in Proc. Int. Conf. Neural Inf. Process. Syst., (New Orleans, LA, USA), Nov. 2022, pp. 24824-24837.

[26] H. Wang et al., "Bitnet: Scaling 1-bit transformers for large language models," arXiv preprint, 2023. doi: 10.48550/arXiv.2310.11453