모바일 LLM 추론을 위한 배치 적응형 GEMM 커널 설계 및 최적화

김종집O, 신동군

성균관대학교 소프트웨어학과

jongjip@g.skku.edu, dongkun@skku.edu

Design and Optimization of an Adaptive GEMM Kernel for Mobile LLM Inference

Jongjip KimO, Dongkun Shin

Department of Computer Science and Engineering, Sungkyunkwan University

요 약

최근 대형 언어모델(LLM)의 온디바이스 실행 수요가 증가함에 따라, 모바일 GPU 환경에서의 효율적인 추론이 중요해지고 있다. 특히 speculative decoding과 같은 추론 가속 기법에서는 작은 크기 배치(N=2-16)의 행렬 곱 연산(GEMM)이 반복 수행되는데, 기존 커널은 특정 배치 구간에 최적화된 정적 전략을 사용하여 성능 편차가 크다. 본 연구에서는 배치 크기에 따라 K 축 분할 비율을 동적으로 조정하는 adaptive split-K 기반 GEMM 커널을 제안한다. Qualcomm Adreno GPU를 타겟으로 워크그룹 구성을 최적화하여, 작은 배치(N=2-16) 구간에서 기존 llama.cpp 대비 최대 3배의 성능 향상을 달성하였다. 특히 Speculative decoding 환경에서 verification 단계 지연시간과 draft 단계 지연시간에서 최대 47% 감소시켜, 다양한 배치 크기 환경에서 일관된 고성능 추론이 가능함을 실험적으로 확인하였다.

1. 서 론[1]

최근 대형 언어모델(LLM, Large Language Model)의 온디바이스(on-device) 실행 수요가 증가함에 따라, 모바일 및 엣지 디바이스 환경에서 효율적인 추론을 위한 연구가 활발히 진행되고 있다. MLC-LLM[1]과 llama.cpp[2]와 같은 오픈소스 추론 엔진은 OpenCL 기반 GPU 가속을 통해 제한된 자원에서도 LLM 구동을 가능하게 한다.

LLM 추론에서 가장 많은 연산량을 차지하는 행렬 곱 연산(GEMM)은 추론 단계에 따라 배치 크기(batch size)가 크게 변한다. Prefill 단계는 입력 토큰 수만큼 큰 배치(N>32)로 수행되고, decoding 단계는 토큰을 하나씩 생성하며 작은 배치(N=1)로 수행된다. 최근 주목받는 speculative decoding[3,4]과 같은 추론 가속 기법에서는 중간 크기 배치(N=4-16)의 GEMM이 반복적으로 발생한다. 따라서 모바일 환경에서 이와 같은 기법을 사용하여 효율적인 LLM 추론을 위해서는 다양한 배치 크기에서 일관된 연산 효율을 유지하는 것이 핵심이다.

그러나 기존 추론 엔진의 GEMM 커널은 정적 병렬화 전략을 사용하여 이러한 다양한 배치 크기에 대해 효과적으로 처리하지 못한다. 본 연구에서는 배치 크기

* 이 논문은 삼성전자 미래기술육성센터의 지원을 받아 수행된 연구임(과제번호 SRFC-IT2402-15)

변화에 강인한 adaptive split-k GEMM 커널을 제안한다. Split-K[5] 병렬화 기반으로 작은 배치에서도 높은 병렬성을 확보하고, 배치 크기별 최적 워크그룹 구성을 자동 선택하는 메커니즘을 설계하였다. 실험 결과, 작은 배치(N=2~16) 구간에서 GEMM 커널이 llama.cpp의 기존 대비 최대 3배 성능 향상을 달성하였으며, speculative decoding 환경에서 end-to-end latency를 47% 감소시켰다.

2. 배경

2.1 기존 GEMM 커널 구조 및 한계

OpenCL에서 워크아이템(Work-item)은 GPU의 기본 실행 단위로, 하나의 스레드에 해당하며 행렬 곱에서 작은 연산 단위를 담당한다. 여러 워크아이템은 워크그룹(Work-group)으로 묶여 동일한 로컬 메모리를 공유하며, 협력하여 더 큰 연산 블록을 수행한다.

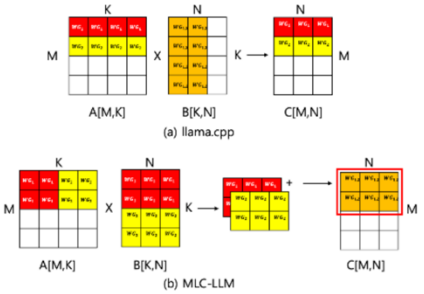

그림 1. llama.cpp(a)와 MLC-LLM(b) 의 GEMM커널 구조 차이

LLM에서 GEMM은 C[M×N] = A[M×K] × B[K×N] 형태의 행렬 곱으로 표현되며, M은 출력 차원, N은 배치 크기, K는 reduction 차원을 의미한다. 그림 1에서 색상(빨강·노랑)은 각각 하나의 워크그룹이 담당하는 연산 영역을 나타낸다. (a)의 llama.cpp 커널은 M축을 기준으로 워크그룹을 분할하여, 각 워크그룹이 K축 전 구간을 순차적으로 처리한다. 이 방식은 큰 배치(N > 32)에서는 각 워크그룹에 충분한 계산량이 할당되어 높은 자원 활용률을 보이지만, 작은 배치(N ≤ 16)에서는 워크그룹당 연산량이 적어 병렬 효율이 저하된다. 반면 (b)의 MLC-LLM은 K축을 일정 비율(예: K/2)로 분할하여 여러 워크그룹이 서로 다른 K 구간을 병렬 계산한 뒤, 부분 결과를 합산하는 split-K 병렬화를 수행한다. 이 방식은 작은 배치에서 병렬성을 높이는 장점이 있으나, 큰 배치에서는 부분 합산(reduction) 오버헤드가 커져 성능이 감소한다. 결국 두 커널 모두 특정 배치 범위에 최적화된 정적 분할 방식을 사용하므로, 다양한 배치 환경에서 일관된 성능을 유지하기 어렵다.

2.2 Speculative Decoding

Speculative decoding은 경량 draft 모델이 여러 후보 토큰을 생성하고, 대형 target 모델이 이를 병렬로 검증하는 2단계 구조로 디코딩 속도를 개선한다. Draft 단계에서는 트리 구조와 토큰의 확률 분포에 따라 배치 크기가 동적으로 변한다(batch size=1-8). Verification 단계에서는 생성된 토큰 트리 전체를 한 번에 처리하므로 트리 크기가 배치 크기로써 GEMM 연산이 수행된다. 이처럼 다양한 배치 크기의 GEMM이 반복적으로 발생한다.

3. 커널 설계

2장에서 분석한 바와 같이 기존 GEMM 커널은 특정 배치 구간에 최적화된 정적 병렬화 전략을 사용하여 다양한 배치 환경에서 일관된 성능을 유지하지 못한다. 본 연구에서는 llama.cpp의 OpenCL 기반 GEMM 커널을 기반으로, 배치 크기에 따라 K축 분할 비율을 동적으로 조정하는 adaptive split-K 커널을 제안한다.

3.1 Adaptive Split-K 커널 구조

이 논문에서 제안하는 커널은 GEMM 연산을 워크그룹과 워크아이템 계층으로 분할하여 병렬 수행한다. 출력 행렬 C[M×N]은 (WI_M × WI_N × WI_K) 크기의 타일로 나뉘며, 워크그룹은 각각 M/WI_M, N/WI_N, K/WI_K 개로 생성된다. 각 워크그룹 내부에서는 WI_M × WI_N × WI_K 개의 워크아이템이 협력하여 타일 단위의 부분 합을 계산하고, 최종적으로 전체 결과를 병합한다. 이러한 계층적 분할 구조를 통해 M, N, K 세 방향의 병렬성을 동시에 활용할 수 있으며, 다양한 배치 크기에서도 연산 자원을 효율적으로 활용할 수 있다.

```
for each tile_m in 0 .. M / WI_M - 1:
    for each tile_n in 0 .. N / WI_N - 1:
        for each tile_k in 0 .. K / WI_K - 1:
            for wm = 0 to WI_M - 1:
                for wn = 0 to WI_N - 1:
                    for wk = 0 to WI_K - 1:
                        i = tile_m * WI_M + wm
                        j = tile_n * WI_N + wn
                        k = tile_k * WI_K + wk
                        C[i, j] += A[i, k] * B[k, j]
```

그림 2. GEMM 커널 타일링 연산 Pseudocode

3.2 배치 크기별 타일링 파라미터 최적화

최적의 (WI_M, WI_N, WI_K) 조합은 배치 크기와 하드웨어 특성에 따라 달라진다. 본 연구에서는 Qualcomm Adreno GPU를 대상으로, 해당 아키텍처의 워크그룹 크기를 Qualcomm에서 권장하는 서브그룹(워크그룹 단위) 크기인 128을 기준으로 autotuning을 수행하였다. 탐색 공간은 WI_M, WI_N, WI_K를 각각 2의 거듭제곱 값으로 설정하고, 세 값의 곱이 128이 되도록 조합을 구성하였다. 배치 크기(batch size=2~256)에 대해 각 조합의 실행 시간을 OpenCL

이벤트 타이밍으로 측정하여 최대 처리량을 달성하는 구성을 최적 파라미터로 선정하였다.
튜닝 결과, 작은 배치에서는 WI_K가 상대적으로 커지고, 큰 배치에서는 WI_K가 작게 설정되는 경향을 보였다. 이는 배치가 작을수록 K축을 더 세분화하여 병렬성을 극대화하고, 반대로 배치가 커질수록 reduction 연산의 오버헤드를 최소화하는 방향으로 최적화가 이루어진 것이다. 도출된 파라미터 조합은 커널 내부 테이블에 저장되며, 런타임에서 입력 배치 크기에 따라 자동으로 선택되어 실행된다. 이를 통해 배치 크기가 동적으로 변하는 환경에서도 일관된 고성능을 유지할 수 있다.

4. 실험 환경 및 결과

4.1 실험 환경

본 실험은 Samsung Galaxy S25(Snapdragon 8 Elite 칩셋) 환경에서 진행하였다. MLC-LLM과의 Prefill 성능 비교에서는 MLC-LLM이 지원하는 모델인 llama-3.2-3B-Instruct(Q4_0 정밀도)를 사용하였고, Speculative Decoding 실험에서는 Target Model은 Vicuna 7B (Q4_0 정밀도)를 사용하였으며, Draft Model은 Eagle Layer(Q4_0 정밀도)을 사용하였다.
성능 측정은 OpenCL 이벤트 타이밍으로 커널 레벨 TFLOPS과 Prefill Throughput(tok/s)를 측정하였고, End to End관점에서 speculative decoding의 verification latency와 draft latency를 측정하였다.

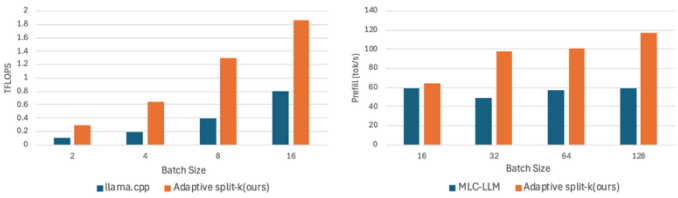

그림 3. (a) 배치 크기에 따른 GEMM 커널 성능(TFLOPS) 비교: llama.cpp vs. Adaptive split-K
(b) 배치 크기에 따른 Prefill 처리 성능(tok/s) 비교: MLC-LLM vs. Adaptive split-K

4.2 성능 비교

그림 3은 다양한 배치 크기구간에서 제안한 Adaptive split-K 기반 커널의 성능을 기존 구현과 비교한 결과이다. [그림 3-a]는 llama.cpp의 기본 GEMM 커널과 제안된 커널의 TFLOPS 성능을 비교한 것으로, 기존 커널은 작은 배치(batch size=2-16)에서는 자원 활용률이 낮은 한계를 보인다. 반면 제안된 커널은 split-K 병렬화를 통해 이 구간에서 최대 3배의 성능을 달성했다. [그림 3-b]는 MLC-LLM과 제안된 커널의 성능을 입력의 토큰 수가 배치 사이즈로서 수행되는 Prefill에서 큰 배치 구간(batch size=16-128)에서 Throughput(tok/s)를 비교한 결과로, 제안된 커널이 MLC-LLM에 비해 큰 배치사이즈에서도 안정적으로 좋은 성능을 보이고 있다.

4.3 Speculative Decoding의 성능 비교

표 1은 Draft Tree Size = 8 환경에서 speculative decoding 수행 시 각 단계의 지연 시간을 비교한 결과이다. 제안한 커널을 적용한 경우, draft 단계(batch size = 1-4) 지연은 약 17.8 %, verification 단계 지연은 약 47.3 % 감소하였다. 특히 verification 단계는 8 배치(batch size = 8) 단위의 GEMM 연산이 반복적으로 수행되는데, 이는 제안된 커널이 작은 배치 구간(N ≤ 16)에서 최적화되어 높은 처리 효율을 보이는 영역과 일치한다.

표 1. Speculative Decoding에서의 Draft, Verification Latency 속도 측정(Target Model : Vicuna 7B_Q4_0 Draft Model : Eagle Layer_Q4_0)

ms	llama.cpp	Adaptive split-k
Draft	27.246	22.387
Verification	224.795	118.374

5. 결론

본 연구에서는 모바일 GPU 환경에서 배치 크기 변화에 강인한 adaptive split-K 기반 GEMM 커널을 설계하였다. Qualcomm Adreno GPU의 하드웨어 특성을 고려한 워크그룹 구성 최적화를 통해 작은 배치(N=2~16) 구간에서 최대 3배의 성능 향상을 달성하였다. 또한 speculative decoding 환경에서 verification 단계의 지연 시간을 47% 감소시켜, 배치 크기가 동적으로 변하는 모바일 LLM 추론에서 효율적인 가속이 가능함을 실험적으로 확인하였다.

참고 문헌

[1] https://github.com/mlc-ai/mlc-llm
[2] https://github.com/ggml-org/llama.cpp
[3] Yaniv Leviathan, Matan Kalman, Yossi Matias, “Fast Inference from Transformers via Speculative Decoding,” Proceedings of the 40th International Conference on Machine Learning (ICML 2023), Vol. 202, pp. 19274–19286, 2023.
[4] Li, Y., Wei, F., Zhang, C., Zhang, H., EAGLE: Speculative Sampling Requires Rethinking Feature Uncertainty, International Conference on Machine Learning (ICML), 2024.
[5] NVIDIA CUTLASS Documentation, "Efficient GEMM in CUDA, https://docs.nvidia.com/cutlass/media/docs/cpp/efficient_gemm.html