SLO 인지 Prefill 스케줄링을 통한 CXL-PIM 기반 LLM 추론 성능 개선

*이성훈, 정의영
연세대학교 전기전자공학부 소속
e-mail : crospham1@yonsei.ac.kr, eychung@yonsei.ac.kr

SLO-Aware Prefill Scheduling for Improving CXL-PIM LLM Inference

*SungHoon Lee and Eui-Young Chung
Yonsei University Department of Electrical and Electronic Engineering

Abstract

Large Language Model (LLM) inference is heavily constrained by memory bandwidth due to massive parameter and key-value cache accesses. In multi-tenant environments, requests with diverse lengths and latency targets arrive concurrently, and the way contention between prefill and decode stages is managed strongly affects average response time and SLO violation rate. This paper proposes WASP (Work-Aware Scheduling for Prefill), a prefill scheduling policy for a simulated CXL-PIM-based LLM inference environment, which prioritizes requests according to their remaining work. WASP combines a work-estimation model that accounts for prefix sharing and the benefits of speculative decoding, chunked prefill execution, and slack-aware emergency prioritization to mitigate head-of-line blocking and excessive delay of long prefill requests. To evaluate the design, we implement a discrete-event simulator that models a CXL-PIM-based LLM inference system and compare WASP against FIFO and the MultiTier scheduling baselines. Under the heavy-load scenario, WASP reduces average response time and SLO violation rate compared with FIFO and MultiTier, while alleviating queue congestion. However, WASP shows a higher P99 latency than MultiTier because a small number of long prefill requests can be delayed by work-aware prioritization and repeated chunk re-insertion. These results reveal a trade-off between average-case performance and tail latency.

I. 서론

대규모 언어 모델(LLM) 추론은 긴 입력 전체를 처리하여 초기 KV cache를 생성하는 프리필(prefill) 단계와, 이후 토큰을 순차적으로 생성하는 디코딩(decode) 단계로 구성된다. 프리필은 입력 길이에 비례해 큰 메모리 접근 비용을 요구하고, 디코딩은 반복적인 KV cache 접근을 수반하므로, 멀티테넌트 환경에서는 요청 간 자원 경쟁이 평균 응답 시간, 꼬리 지연, SLO(Service Level Objective) 위반율에 큰 영향을 미친다[1]. 특히 긴 프리필 요청이 먼저 실행되면 뒤따르는 요청이 함께 지연되는 Head-Of-Line(HOL) 블로킹이 발생할 수 있고, 반대로 디코딩을 지속적으로 우선하면 긴 프리필 요청의 완료가 과도하게 늦어질 수 있다. 그러나 기존

CXL-PIM 기반 LLM 추론 연구는 주로 시스템 구조와 실행 환경에 초점을 두고 있으며, 다양한 요청이 혼재하는 환경에서 이러한 경쟁을 조정하는 소프트웨어 수준 스케줄링은 충분히 다루어지지 않았다.

본 논문은 이러한 문제를 해결하기 위해, CXL-PIM 기반 LLM 추론 환경을 반영한 시뮬레이션 모델에서 프리필 요청의 잔여 작업량을 기준으로 우선순위를 부여하는 WASP(Work-Aware Scheduling for Prefill)를 제안한다. WASP는 prefix sharing과 speculative decoding의 이점을 반영한 작업량 추정, 분할 가능한 프리필 실행, 그리고 SLO slack 기반 긴급 우선 처리를 결합하여 긴 프리필 요청으로 인한 HOL 블로킹과 과도한 지연을 완화한다. 이를 평가하기 위해 CXL-PIM 기반 LLM 시스템 구성을 반영한 이산 사건 시뮬레이터를 구현하고, FIFO 및 MultiTier 스케줄링과 비교하였다. 실험 결과, WASP는 과부하 환경에서 평균 응답 시간과 SLO 위반율을 줄이지만, 일부 긴 요청의 지연 증가로 인해 전체 P99 latency는 악화될 수 있음을 확인하였다. 이는 평균 성능과 tail latency 사이의 trade-off를 보여준다.

논문의 기여는 다음과 같다. 첫째, 멀티테넌트 LLM 추론 환경에서 프리필과 디코딩 간 경쟁이 평균 응답 시간과 SLO 위반율에 미치는 영향을 분석한다. 둘째, 잔여 작업량 기반 우선순위, 분할 가능한 프리필 실행, SLO slack 기반 긴급 우선 처리를 통합한 WASP를 제안한다. 셋째, CXL-PIM 기반 LLM 추론 시스템을 반영한 이산 사건 시뮬레이터를 통해 WASP의 효과와 한계를 함께 평가한다.

II. 제안 기법

2.1 WASP 개요

멀티테넌트 LLM 추론 환경에서는 프리필과 디코딩 요청이 동일 자원을 두고 경쟁하므로, 단순 FIFO나 디코딩 우선 정책만으로는 HOL 블로킹, 긴 프리필 요청의 과도한 지연, 그리고 높은 SLO 위반율을 동시에 완화하기 어렵다. 이를 해결하기 위해 본 논문은 프리필 요청의 잔여 작업량을 기준으로 우선순위를 결정하는 WASP(Work-Aware Scheduling for Prefill)를 제안한다. WASP는 work-aware 우선순위, chunked prefill 실행, 그리고 SLO slack 기반 긴급 우선 처리를 결합하여 평균 응답 시간과 SLO 위반율을 낮추는 것을 목표로 한다. 구체적으로 WASP는 각 요청의 입력 길이뿐 아니라 예상 출력 길이, prefix sharing, speculative decoding 효과를 함께 고려하여 남은 작업량을 추정한다. 또한 긴 프리필 요청을 청크 단위로 분할 실행함으로써 단일 요청이 파이프라인을 장시간 점유하여 후속 요청을 지연시키는 상황을 완화한다. 마지막으로 SLO slack이 부족한 요청을 긴급 큐에서 우선 처리하여, 평균 응답 시간 개선 과정에서 마감이 임박한 요청이 과도하게 지연되지 않도록 한다.

Algorithm 1 Work-Aware Prefill Priority

```
function OnRequestArrival(req)
    compute T_exp(req); set arrival_time, rem_pre
    insert req into Q_pre ordered by T_exp
end function
```

그림 1. Work-Aware Prefill Prioritization

2.2 Work-Aware Prefill 우선순위

WASP는 프리필 큐에서 요청을 선택할 때 단순 입력 길이 대신 예상 잔여 작업량을 사용한다. 요청 i의 총 예상 실행 시간 T_{exp}는 다음과 같이 정의한다.

$$T_{exp} = (1-h) \times L_{prefill} \times \tau_{prefill} + \frac{L_{decode}}{g} \times \tau_{decode}$$

여기서 $L_{prefill}$은 남은 프리필 토큰 수, L_{decode}는 예상 디코딩 토큰 수, h는 prefix sharing에 따른 유효 프리필 감소 비율, g는 speculative decoding 이득이다[2],[3]. $\tau_{prefill}$과 τ_{decode}는 각각 토큰당 프리필 및 디코딩 지연 시간이다. WASP는 T_{exp}가 작은 요청을 우선 선택함으로써, 실제 완료 시간이 짧을 가능성이 높은 요청을 먼저 처리한다[4].

Algorithm 2 Chunked Prefill Execution

```
function ExecuteTask(task, type)
    if type == DECODE then
        run one decode chunk
        if remaining then
            enqueue Q_dec
        else
            complete
        end if
    else                                          ▷ PREFILL
        n ← min(rem_pre(task), T_chunk)
        run one prefill chunk of n
        update rem_pre(task)
        if rem_pre(task) > 0 then
            enqueue Q_pre ordered by T_exp
        else
            enqueue Q_dec
        end if
    end if
end function
```

그림 2. Chunked Prefill Execution

2.3 Chunked Prefill 실행

긴 프리필 요청은 한 번에 끝까지 실행될 경우 파이프라인을 장시간 점유하여 HOL 블로킹을 유발할

수 있다. 이를 완화하기 위해 WASP는 프리필을 고정 길이 청크 단위로 나누어 실행한다[5]. 각 요청은 한 번의 스케줄링에서 고정된 크기의 프리필 구간까지만 수행하고, 남은 프리필 토큰이 있으면 다시 큐에 삽입된다. 이를 통해 긴 요청도 주기적으로 실행 기회를 양보하게 되어 짧은 요청과 긴급 요청이

Algorithm 3 SLO-Aware Urgent Processing

```
function ScheduleNextTask
    urgent ← arg min_{r∈Q_pre} slack(r)  with  slack(r) < Slack_thresh
    if urgent exists then
        return (urgent, PREFILL)
    else if Q_dec ≠ empty then
        return (pop_front(Q_dec), DECODE)
    else if Q_pre ≠ empty then
        return (pop_min_Texp(Q_pre), PREFILL)
    else
        return None
    end if
end function
```

그림 3. SLO-Aware Emergency Prioritization

중간에 개입할 수 있다.

2.4 SLO-Aware 긴급 우선 스케줄링

WASP는 각 요청의 남은 SLO slack을 계산하고, 이 값이 임계값 이하로 감소한 프리필 요청을 긴급 큐로 이동시킨다. 스케줄러는 긴급 프리필, 디코딩, 일반 프리필 순으로 작업을 선택하여 마감이 임박한 요청을 우선 보호한다. 이를 통해 디코딩 우선 정책에서 발생할 수 있는 긴 프리필 요청의 과도한 지연과 높은 SLO 위반율을 완화한다[4],[6].

Ⅲ. 실험 및 평가

3.1 시스템 모델 및 시뮬레이터 구현

본 연구에서는 CXL-PIM 기반 LLM 추론 환경을 반영한 논리적 성능 모델을 사용한다[7]. 기존 CXL-PIM LLM 시스템의 공개 결과를 바탕으로 토큰당 지연 시간과 처리량 특성을 추출하고, 이를 룩업 테이블 형태로 구성하였다. Python으로 구현한 이산 사건 시뮬레이터는 각 요청의 프리필 및 디코딩 진행을 토큰 단위로 모델링하며, 파이프라인 슬롯 수는 32 디바이스 환경에 맞춰 고정하였다.

3.2 실험 설정

본 연구는 trace-driven 이산 사건 시뮬레이션을 통해 FIFO, MultiTier, WASP를 동일한 조건에서 비교한다. 본 논문에서 MultiTier는 프리필 요청과 디코딩 요청을 별도 큐로 분리하고, 상대적으로 처리 시간이 짧은 디코딩 요청을 항상 우선 처리하는 multi-queue 스케줄링으로 정의하며, 마감 시간(deadline)은 고려하지 않는다. 입력 트레이스는 실제 서비스 로그가 아니라, 멀티테넌트 LLM 추론 환경에서 길이와 지연 요구가 서로 다른 요청이 혼재하는 상황을 모사하기 위해 생성한 합성 워크로드 트레이스(synthetic workload trace)이다. 각 트레이스는 요청 도착 시간, 입력 토큰 수, 예상 출력 토큰 수, SLO 조건으로 구성되며, 요청 도착 시간은 포아송 도착 과정(Poisson process)을 기반으로 생성하였다. 토큰당 실행 지연은 CXL-PIM 기반 실행 환경의 선행 트레이스 평가 결과를 사용하였다. 32 디바이스 환경에서 프리필 지연은 19.42 ms/token, 디코딩 지연은 25.03 ms/token으로 설정하였다. 부하 조건은 트레이스의 평균 요청 도착률을 조절하여 Light, Medium, Heavy로 구성하였으며, 모든 스케줄러는 동일한 트레이스 세트를 입력받아 평균 응답 시간, P99 latency, SLO 위반율을 기준으로 평가하였다.

3.3 평가 결과

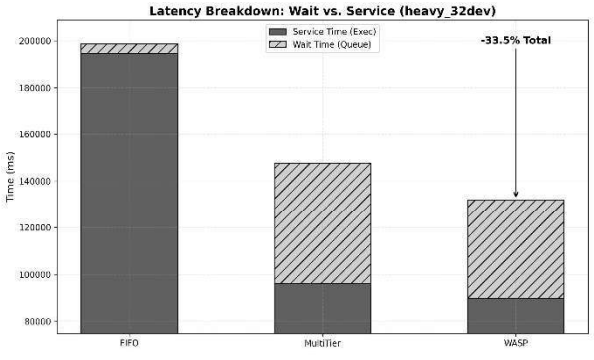
이하의 표 1 및 그림 4, 5는 모두 동일한 heavy_32dev 시나리오에서 측정한 결과이다. heavy_32dev는 32개 디바이스 환경에서 단위 시간당 요청 도착률(Arrival Rate)을 시스템의 최대 처리 용량보다 높게 설정한 과부하 조건으로, 스케줄러 간 차이가 가장 뚜렷하게 나타나는 주요 실험 설정이다.

Table 1: Performance comparison of scheduling policies

Policy	Avg Response Time (s)	P99 Latency (s)	SLO Violation Rate (%)
FIFO	198.7	383.9	90.3
MultiTier	147.6	330.1	77.3

표 1. Performance Comparison

표 1은 각 스케줄러의 평균 응답 시간, P99 latency, SLO 위반율을 비교한 결과를 보여준다. WASP는 FIFO 및 MultiTier보다 평균 응답 시간과 SLO 위반율을 줄였다. 다만 P99 지연 시간은 FIFO보다 낮지만 MultiTier보다 높게 나타났으며, 이는 일부 긴 프리필 요청이 후순위 처리와 청크 단위 재삽입의 영향을 받아 완료가 늦어졌기 때문으로 해석된다. 따라서 WASP는 다수 요청의 평균 대기 시간을



줄이는 반면, 일부 긴 요청의 tail latency를 증가시키는 trade-off를 보인다.

그림 4는 평균 지연 시간을 대기 시간(Wait)과 서비스 시간(Service)으로 분해한 결과이다. FIFO는 긴 프리필 요청에 의한 HOL 블로킹으로 대기 시간이 크게 증가하며, MultiTier는 이를 일부 줄이지만 프리필 큐 적체를 충분히 완화하지 못한다[8]. 반면 WASP는 잔여 작업량 기반 우선순위와 chunked prefill 실행을 통해 다수 요청의 대기 시간을 줄여 전체 평균 응답 시간을 개선한다. 다만 이 과정에서 일부 긴 프리필 요청의 완료는 더 늦어질 수 있어, tail latency 측면에서는 trade-off가 발생한다.

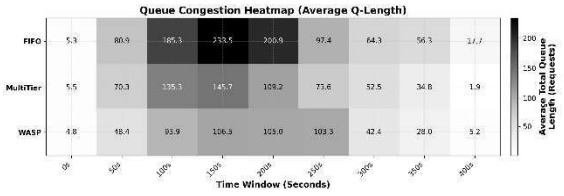


그림 5는 각 스케줄러의 50초 구간별 평균 총 큐 길이(프리필 및 디코딩 대기 요청의 합)를 나타낸 히트맵이다. 색이 짙을수록 해당 시간 구간에서 평균적으로 대기 중인 요청 수가 많아 시스템 혼잡도가 높음을 의미한다. FIFO는 100–200초 구간에서 혼잡이 뚜렷하게 집중되며, 특히 150초 구간에서 평균 총 큐 길이가 233.5로 가장 크게 나타난다. MultiTier는 FIFO의 해당 최대값을 145.7까지 낮추지만, 동일한 과부하 구간에서 높은 혼잡이 여전히 관찰된다. 반면 WASP는 구간 평균 총 큐 길이의 최대값을 106.5로 더 낮추며, 고혼잡 구간의 강도와 지속시간을 모두 완화하는 경향을 보인다. 이는 WASP가 과부하 조건에서 큐 적체가 특정 시점에 집중적으로 누적되는 현상을 완화함을 시사하며, 앞선 표 1과 그림 4에서 확인된 평균 응답 시간 및 대기 시간 개선과도 일관된 결과이다.

그림 5. Queue Congestion over Time

Ⅳ. 결론 및 향후 연구 방향

본 논문은 멀티테넌트 LLM 추론 환경에서 프리필과 디코딩 간 자원 경쟁으로 인해 발생하는 HOL 블로킹, 긴 프리필 요청의 과도한 지연, 그리고 높은 SLO 위반율 문제를 분석하고, 이를 완화하기 위한 WASP를 제안하였다. WASP는 prefix sharing과 speculative decoding의 이점을 반영한 작업량 기반 우선순위, chunked prefill 실행, 그리고 SLO slack 기반 긴급 우선 처리를 결합한다. heavy_32dev 과부하 시나리오에서 WASP는 FIFO 및 MultiTier 대비 평균 응답 시간과 SLO 위반율을 낮추었고, 큐 적체가 특정 시간 구간에 집중되는 현상도 완화하였다. 다만 일부 긴 프리필 요청은 청크 단위 실행과 재삽입 과정에서 완료가 지연되었으며, 그 결과 MultiTier 대비 P99 지연 시간이 증가하는 trade-off가 나타났다. 이는 WASP가 평균 성능과 SLO 준수율을 개선하는 반면, tail latency 측면에서는 추가적인 보완이 필요함을 보여준다.

향후에는 WASP를 여러 요청 분포와 서비스 시나리오에서 검증하고, 청크 크기, slack 임계값, 작업량 추정 방식과 같은 주요 설계 요소가 성능에 미치는 영향을 추가로 분석할 예정이다. 또한 prefix sharing과 speculative decoding 효과를 보다 정교하게 반영할 수 있는 모델로 확장하고, 요청 특성과 서비스 정책을 함께 고려하는 방향으로 스케줄링 기법을 발전시키고자 한다. 나아가 실제 LLM 서빙 환경에 가까운 실행 조건에서 WASP의 동작 특성을 검토하고, 평균 지연, tail latency, SLO 준수 간 균형을 안정적으로 조정하는 후속 연구를 진행할 예정이다.

Ⅴ. 사사

이 논문은 2024년도 산업통상자원부 및 한국산업기술기획평가원(KEIT) (민관공동투자반도체고급인력양성) 사업의 연구비 지원에 의한 연구임('RS-2024-00420541')

참고문헌

[1] Y. Zhong et al., "DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving," in OSDI, 2024.

[2] Y. Leviathan et al., "Fast Inference from Transformers via Speculative Decoding," in ICML, 2023.

[3] X. Miao et al., "SpecInfer: Accelerating Generative Large Language Model Serving with Tree-based Speculative Inference and Verification," in ASPLOS, 2024.

[4] J. Huang et al., "SLO-Aware Scheduling for Large Language Model Inferences," arXiv, 2025.

[5] A. Agrawal et al., "Taming Throughput-Latency Tradeoff in LLM Inference with Sarathi-Serve," in OSDI, 2024.

[6] Y. Sun et al., "Multi-stage Flow Scheduling for LLM Serving," arXiv, 2026.

[7] Y. Gu et al., "PIM Is All You Need: A CXL-Enabled GPU-Free System for Large Language Model Inference," in ASPLOS, 2025.

[8] G.-I. Yu et al., "Orca: A Distributed Serving System for Transformer-Based Generative Models," in OSDI, 2022.