Cross-Point Scheduling(CPS): LLM 서빙 환경에서의 비용 역전 지점 기반 동적 KV 캐시 복구 스케줄링

공경민[O1], 장성[1], 김기현[1], 김영재[†1]

[1]서강대학교 컴퓨터공학과

{kongmin00, jesjsjes, kion777, youkim}@sogang.ac.kr

CPS: Cross-Point Scheduling for Dynamic KV Cache Recovery in LLM Serving

Kyongmin Kong[O1], Seong Jang[1], Kihyun Kim[1], Youngjae Kim[†1]

[1]Department of Computer Science and Engineering, Sogang University

요 약

대규모 언어 모델(LLM) 서빙 환경에서는 긴 컨텍스트와 동시 요청 증가로 인해 GPU 메모리가 부족해지고, 이로 인해 처리 중인 요청이 선점(Preemption)될 수 있다. 기존 vLLM은 선점 복구 시 Swap 또는 Recompute 중 단일 정적 정책을 적용하여 요청 길이에 따른 비용 차이를 충분히 반영하지 못한다. 본 논문은 PCIe 4.0 환경에서 두 복구 방식의 지연 시간을 프로파일링하고, 약 1,024 토큰 부근에서 비용이 역전되는 Cross-Point를 도출하였다. 이를 기반으로 입력 길이에 따라 복구 방식을 동적으로 선택하는 CPS(Cross-Point Scheduler)를 제안한다. 실험 결과, CPS는 평균 TTFT와 평균 TPOT를 모두 개선하였으며, P99 TPOT를 Swap 대비 13.1%, Recompute 대비 20.1% 단축하였다.

1. 서 론

대규모 언어 모델(LLM)의 서비스가 대중화됨에 따라, 실제 서빙 환경에서의 응답 지연 최소화와 자원 활용 효율성이 중요한 문제로 대두되고 있다. 특히 LLM 추론 과정에서는 입력 프롬프트와 생성 토큰에 대한 KV 캐시가 GPU 메모리에 누적되므로, 동시 요청이 증가하거나 긴 컨텍스트가 유입될 경우 가용 메모리 고갈로 인한 선점(Preemption) 현상이 빈번하게 발생한다.

선점된 요청을 복구하는 대표적인 메커니즘은 KV 캐시를 CPU로 오프로딩하는 Swap과 처음부터 다시 연산하는 Recompute이다. 두 방식은 요청의 입력 길이에 따라 성능적 우위가 변하지만 vLLM을 비롯한 기존 서빙 시스템들은 주로 단일 정적 정책을 적용하여 다양한 길이의 요청이 혼재하는 실제 서비스 환경에서 불필요한 오버헤드와 응답 지연을 초래한다.

본 논문은 PCIe 4.0 기반 환경에서 두 방식의 오버헤드를 정량적으로 분석하고, 복구 비용이 역전되는 지점인 비용 역전 지점(Cross-Point)를 도출한다. 나아가 Cross-Point를 기준으로, 런타임 입력 길이에 맞춰 복구 방식을 선택하는 CPS(Cross-Point Scheduler) 기법을 제안한다. vLLM에 CPS를 통합하여 실험한 결과, 기존 방식 대비 평균 초기 응답 시간(Mean TTFT)을 가장 낮은 수준으로 단축하였으며, 특히 시스템의 병목을 나타내는 최악의 1% 꼬리 지연 시간(P99 TPOT)을 최대 20.1% 개선하였다.

2. 배경 지식 및 연구 동기

2.1 LLM 추론 단계: Prefill 및 Decode

LLM 추론은 크게 Prefill과 Decode 단계로 구분된다. Prefill 단계에서는 입력 프롬프트 전체를 병렬로 처리하여 첫 번째 출력 토큰을 생성하고, 입력 토큰에 대한 KV 캐시를 GPU 메모리에 저장한다. Decode 단계에서는 이전 토큰을 기반으로 다음 토큰을 순차적으로 생성하며, 생성이 진행될수록 요청당 KV 캐시 크기가 선형적으로 증가한다.

* 본 연구는 2026년 과학기술정보통신부 및 정보통신기획평가원의 AI중심대학사업 지원을 받아 수행되었음(2026-0-00036).

† Corresponding author

2.2 선점(Preemption) 발생 및 복구 메커니즘

LLM 서빙 환경에서는 KV 캐시 증가로 인해 가용 메모리 블록이 부족해질 수 있다. 시스템은 OOM을 방지하기 위해 일부 요청을 일시적으로 중단하는데, 이를 선점(Preemption)이라고 한다. 대표적인 서빙 프레임워크인 vLLM은 최근에 큐에 진입한 요청을 먼저 선점하는 LIFO(Last-In-First-Out) 정책을 기본적으로 사용한다.

선점된 요청은 점유하던 GPU 자원을 즉각 반납하고 대기 큐로 이동하며, 이후 메모리가 확보되면 Swap 또는 Recompute 방식으로 복구된다. Swap은 KV 캐시를 CPU 메모리로 이동한 뒤 필요 시 다시 GPU로 복원하는 방식으로 PCIe 대역폭에 의존한다. Recompute는 기존 KV 캐시를 폐기하고 처음부터 재연산하는 방식으로, 컨텍스트가 길어질수록 GPU 연산 비용이 증가한다.

2.3 관련 연구 및 본 연구의 차별성

vLLM[1]을 비롯한 기존 LLM 서빙 시스템은 선점 복구 시 컨텍스트 길이와 관계없이 Swap 또는 Recompute 중 하나의 정적 정책을 사용한다. 이는 요청 길이에 따라 GPU 연산 자원과 PCIe 통신 대역폭 중 어느 자원을 활용하는 것이 유리한지 반영하지 못한다.

이를 개선하기 위해 최근 복구 방식을 동적으로 최적화하려는 연구들[2, 3]이 제안되었다. 그러나 기존 연구들은 주로 고대역폭 통신 환경을 전제하였고, 통신 오버헤드를 데이터 전송량을 대역폭으로 나누는 단순 이론 모델로만 계산하는 한계가 있다.

본 연구는 고성능 GPU와 상대적으로 저속인 PCIe 4.0이 공존하는 하드웨어 자원 비대칭성에 주목한다. 단순 이론적 계산을 넘어 시스템 오버헤드를 포함한 실증적 프로파일링을 수행한 결과, 데이터 전송 비용이 재연산 비용을 상회하는 비용 역전 지점(Cross-Point)이 기존 연구의 이론적 계산과 다르게 형성됨을 확인하였다. 본 연구는 이러한 실제 물리적 병목을 반영하여 Cross-Point를 도출하고, 이를 바탕으로 한 동적 스케줄링 기법 CPS(Cross-Point Scheduler)를 제안한다.

3. CPS(Cross-Point Scheduler) 설계 및 구현

3.1 선점(Preemption) 통제 환경 구축

두 복구 방식의 지연 시간을 정확히 비교하기 위해, 의도적으로 선점을 유발하는 통제 환경을 구축하였다. 전체 KV 캐시 용량을 512 블록으로 고정하고, vLLM의 OOM 방지용 워터마크 정책을 비활성화하였다. 이를 통해 고정된 512 블록이 소진된 직후 새로운 토큰 생성이 요구될 때 선점이 결정적으로 발생하도록 하였다.

3.2 복구 지연 시간(Latency) 프로파일링

3.2.1 Swap 복구 지연 시간

Swap 지연 시간을 스케줄러 준비 시간인 Swap prep, GPU에서 CPU로 이동하는 Swap out, CPU에서 GPU로 복원하는 Swap in으로 나누어 측정하였다. CUDA 이벤트 동기화를 활용해 CPU 명령 디스패치 시간과 PCIe 4.0 기반 데이터 복사 시간을 분리하여 실제 I/O 병목을 정량화하였다.

3.2.2 Recompute 복구 지연 시간

순수한 재연산 비용을 측정하기 위해 상태 추적 플래그를 도입하였다. 선점 발생 시 스케줄러가 해당 요청에 마커를 부여하고, 이를 워커 노드(ModelRunner)에 전달한다. 이후 모델 실행 단계에서 해당 마커를 식별하여 일반 추론과 구분된 Re-Prefill 지연 시간만을 측정하였다.

3.3 Cross-Point 식별 및 스케줄러 통합

위의 프로파일링을 통해 컨텍스트 길이 1,024 토큰 부근을 기준으로 Swap과 Recompute의 복구 비용이 역전되는 교차점(Cross-Point)을 확인하였다. 도출된 이 임계값을 vLLM 스케줄러(scheduler.py) 로직에 직접 통합하여 CPS 기법을 구현하였다. 구체적으로, 선점 대상의 시퀀스 길이를 검사하여 1,024 토큰 이하인 요청에는 Swap을, 이를 초과하는 요청에는 Recompute 정책을 할당하도록 제어 흐름을 수정하였다. 이를 통해 하드웨어의 통신 대역폭과 GPU 연산 능력 사이의 불균형을 완화하였다.

표 1. 하드웨어 및 소프트웨어 실험 환경 세부 설정

Component	Specification
Server Experiment Environment	
CPU	Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz
GPU	NVIDIA RTX A6000 (48GB VRAM)
System Memory	512GB RAM
Interconnect	PCIe 4.0 x 16
vLLM / CUDA	v0.2.4 / 12.1
Experimental Details	
Model	Qwen2.5-0.5B
Model Temperature	1.0 (vLLM default)
Dataset	shareGPT (200 samples)

4. 실험 및 평가

4.1 실험 환경 및 평가 데이터셋 구성

본 연구는 표 1 에 명시된 하드웨어 및 소프트웨어 환경에서 실험을 진행하였다. 실제 서비스 환경 모사를 위해 ShareGPT 데이터셋[4]을 활용하여 프롬프트와 응답 총길이가 128~2,048 토큰인 데이터를 선별하였다. 도출된 임계점의 효과를 검증하기 위해, 이를 기준으로 짧은 요청(Short)과 긴 요청(Long)을 각각 100개씩 무작위 추출하여 총 200개의 혼합 데이터셋을 구축하였다.

극한의 선점 병목 환경을 강제로 유도하기 위해, 3.1절의 통제 환경(KV 캐시 512 블록 고정) 위에서 이 200개의 요청을 최대 유입 속도인 --request-rate inf로 한 번에 투입하여 각 복구 정책의 영향을 관찰하였다.

표 2. 컨텍스트 길이에 따른 선점 복구 메커니즘(Swap 및 Recompute) 지연시간 비교

Input Length	Swapping (ms)				Recomputation (ms)
	Swap OUT	Swap PREP	Swap IN	**Swap Latency**	**Recompute Latency**
512	3.236	0.034	3.366	**6.64**	8.30
1024	6.257	0.050	6.379	12.69	**12.30**
2048	17.795	0.084	18.411	36.29	**22.73**
4096	34.632	0.151	36.658	71.44	**42.90**
8192	68.939	0.281	72.574	141.80	**97.60**

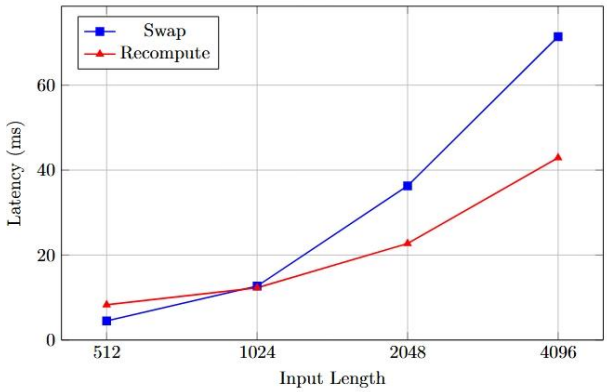

그림 1. 컨텍스트 길이에 따른 복구 전략별 오버헤드 추이 및 비용 교차점(Cross-Point)

4.2 실험 결과 분석

4.2.1 지연 시간 프로파일링 및 Cross-Point 식별

표 2 는 512~8,192 토큰 구간의 복구 지연 시간을 제시하며, 그림 1은 주요 구간에서의 오버헤드 추이를 시각화한다. 짧은 입력에서는 데이터 전송량이 적어 Swap이 낮은 지연을 보였으나, 길이가 증가할수록 I/O 병목으로 인해 전송 비용이 점차 누적되었다. 반면 Recompute는 짧은 입력에서는 불리하지만, 일정 길이 이상에서는 GPU의 병렬 처리 능력 덕분에 재연산 비용 증가율이 상대적으로 낮게 유지되었다.

4.2.2 CPS(Cross-Point Scheduler) 성능 평가

식별된 Cross-Point를 기준으로 CPS의 전체 서빙 성능을 평가하였다. 성능 지표로는 첫 토큰 생성 지연인 TTFT와 이후 토큰당 생성 지연인 TPOT를 사용하였다. 그림 2 에 따르면 CPS는 평균 TTFT 1,648.35 ms, 평균 TPOT 36.55 ms를 기록하여 단일 정적 정책 대비 가장 우수한 성능을 보였다.

특히 시스템 병목을 나타내는 최악의 1% 꼬리 지연 시간인 P99 TPOT의 경우, CPS는 70.57 ms를 기록하여 Swap 대비 13.1%, Recompute 대비 20.1% 단축되었다. 이는 CPS가 요청 길이에 따라 복구 방식을 동적으로 선택함으로써, 짧은 요청에서는 불필요한 재연산 비용을 줄이고 긴 요청에서는 과도한 I/O 전송 비용을 완화했기 때문으로 해석된다. 따라서 CPS는 혼합 길이 워크로드 환경에서 정적 복구 정책보다 낮은 응답 지연을 달성함을 확인하였다.

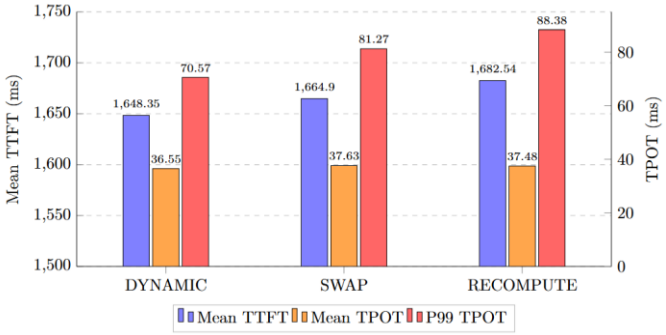

그림 2. 정책별 초기 응답 시간(TTFT) 및 토큰 생성 시간(TPOT) 비교

5. 결 론

본 연구는 LLM 서빙 환경에서 선점 복구 오버헤드를 줄이기 위한 CPS(Cross-Point Scheduler)를 제안하였다. PCIe 4.0 환경에서 Swap과 Recompute의 지연 시간을 프로파일링하고, 두 방식의 비용이 역전되는 Cross-Point를 도출하였다. 이를 기준으로 짧은 요청에는 Swap을, 긴 요청에는 Recompute를 선택하도록 vLLM 스케줄러에 CPS를 구현하였다.

혼합 워크로드 실험 결과, CPS는 P99 TPOT를 Swap 대비 13.1%, Recompute 대비 20.1% 단축하였다. 또한 평균 TTFT와 평균 TPOT에서도 단일 정적 정책보다 낮은 지연 시간을 보여, 하드웨어 병목을 반영한 동적 복구 정책이 LLM 서빙의 응답 지연을 완화할 수 있음을 확인하였다.

향후 연구에서는 하드웨어 스펙, 인터커넥트 대역폭, 모델 크기 변화에 따라 Cross-Point를 자동으로 재탐색하거나, 운영 중 관측되는 Swap/Recompute 지연 시간을 기반으로 임계값을 동적으로 보정하는 온라인 CPS로 확장할 계획이다.

6. 참고 문헌

[1] W. Kwon et al., "Efficient Memory Management for Large Language Model Serving with PagedAttention", Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), pp. 1-17, 2023.

[2] Y. Sheng et al., "FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU", Proceedings of the International Conference on Machine Learning (ICML), pp. 31094-31116, 2023.

[3] Y. Zhong et al., "DistServe: Disaggregating Prefill and Decoding for Goodput-optimized LLM Serving", Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 1-15, 2024.

[4] LMSYS Org., "ShareGPT: Shared Conversations from ChatGPT", Hugging Face Datasets, 2023.