CXL-SSD를 활용한 LMCache KV 캐시 오프로딩 성능 최적화 연구

봉성호[1], 노태완[1], 정현선[1], 김영재[†1]
[1]서강대학교 컴퓨터공학과
{fhtk1313, heize0502, hchung1652, youkim}@sogang.ac.kr

CXL-SSD–Accelerated KV Cache Offloading for LMCache

SeongHo Bong[1], Taewan Noh[1], Hyunsun Chung[1], Youngjae Kim[†1]
[1]Department of Computer Science and Engineering, Sogang University
{fhtk1313, heize0502, hchung1652, youkim}@sogang.ac.kr

요 약

LMCache는 LLM 추론 시 생성되는 KV 캐시를 GPU 외부로 오프로딩하여 쿼리 간 재사용함으로써 중복 연산을 제거하는 시스템이다. 이 LMCache가 효과적으로 동작하기 위해서는 대용량의 빠른 저장 공간이 요구되며, 호스트 DRAM은 높은 비용과 물리적 연결 한계로 인해 충분한 용량 확보가 어렵다. 본 연구는 이러한 DRAM의 한계를 보완할 수 있는 계층으로 CXL-SSD를 제안한다. CXL-SSD는 바이트 단위 접근이 가능한 대용량 메모리 확장 장치로, 상용화된 제품의 부재로 실험적 검증이 어려운 실정이다. 이에 본 연구에서는 실제 CXL-SSD의 동작을 정밀하게 모사하는 에뮬레이터인 Cylon을 활용하여, CXL-SSD를 LMCache의 백엔드 메모리로 사용하였을 때의 성능을 정량적으로 분석한다. 실험 결과, NAND 접근 지연 시간을 효과적으로 은닉할 수 있다면 CXL-SSD가 LMCache의 DRAM 대체 메모리 확장 계층으로 충분히 활용 가능함을 확인하였다.

1. 서론

LLM 추론 시 동일한 프리픽스를 포함하는 쿼리가 반복적으로 입력될 경우, 매번 KV 캐시를 재연산해야 하는 비효율이 발생한다. 최근 이를 해결하기 위해 한 번 생성한 KV 캐시를 GPU 외부에 보관하고 후속 쿼리에서 재사용함으로써 중복 연산을 제거하는 LMCache[1] 시스템이 등장하였다. LMCache의 효과는 외부 저장소에 보존된 KV 캐시가 후속 쿼리에서 얼마나 많이 재사용되는가에 좌우된다. 따라서 외부 메모리의 용량이 부족할 경우 잦은 축출(Eviction)로 보존 가능한 캐시가 줄어들고, 결국 재연산이 다시 발생하여 LMCache의 본래 효과가 크게 떨어진다. 이를 근본적으로 해결하기 위해서는 대용량이면서도 빠른 메모리 계층이 필요하나, DRAM은 높은 비용과 물리적 연결 한계로 인해 충분한 용량 확보가 어렵다.

한편, CXL-SSD는 CXL.mem 프로토콜을 통해 SSD를 바이트 단위로 접근 가능한 메모리 확장 장치로 활용하는 기술로, DRAM 대비 낮은 비용으로 대용량 확장이 가능하다. CXL-SSD는 내부적으로 소용량 DRAM 캐시와 대용량 NAND 플래시로 구성되며, DRAM 캐시 적중 시에는 주 메모리 수준의 빠른 접근 속도를 보인다. 그러나 캐시 미스 시에는 NAND 플래시로 요청이 전달되어 수십 μs에 달하는 접근 지연이 치명적인 병목으로 작용할 수 있다. 이때 NAND 내부에서 발생하는 가비지 컬렉션(GC)과 쓰기 증폭(Write Amplification)은 미스 지연을 더욱 악화시키므로, 데이터 갱신 빈도가 낮고 접근 단위가 커서 이러한 NAND 부담을 본질적으로 줄여줄 수 있는 워크로드와 결합될 때 CXL-SSD의 잠재력이 극대화된다. LMCache의 KV 캐시는 한 번 쓰이고 반복적으로 읽히는 WORM(Write-Once, Read-Many) 패턴과 청크 단위의 순차적 접근 특성을 가지므로, invalid page 생성이 적어 GC 부담이 줄고, 큰 청크 단위 순차 접근이 NAND die/plane 병렬성을 자연스럽게 활용한다. 따라서 CXL-SSD는 LMCache의 I/O 특성과 구조적으로 높은 친화성을 갖는 메모리 확장 계층이 될 수 있다.

본 논문은 CXL-SSD를 LMCache의 메모리 확장 계층으로 제안하고, Cylon 에뮬레이터[2] 환경에서 LMCache Standalone 모드를 활용하여 실제 LLM 추론 없이 KV 캐시 I/O 경로만을 독립적으로 평가함으로써 그 효용성을 검증한다. 실험 결과, DRAM 캐시 적중 구간에서는 호스트 메모리와 유사한 성능을 보였으나, 미스 구간에서는 NAND 접근 지연으로 인한 급격한 성능 저하가 확인되었다. SSD 내부의 프리페치 기법을 통해 미스 구간의 성능을 부분적으로 개선할 수 있었으나, 순수 DRAM 대비 성능 격차는 여전히 존재하였다. 따라서 향후 LMCache의 KV 캐시 저장·관리 방식에 특화된 호스트 수준의 최적화 연구가 필요함을 제안한다.

2. 연구 배경

LMCache 기반 KV 캐시 외부화 및 재사용. LLM 추론의 프리필(Prefill) 단계에서 생성되는 KV 캐시는 본래 해당 쿼리의 디코드(Decode) 단계에만 사용된 뒤 폐기된다. 그러나 시스템 프롬프트나 동일 문서처럼 공통 prefix가 반복되는 워크로드에서는, 이 KV 캐시를 쿼리 수명 너머로 보존했다가 후속 쿼리에 재사용하면 공통 prefix에 대한 prefill 연산을 통째로 생략할 수 있다. LMCache는 이러한 쿼리 간 KV 캐시 재사용(Cross-query KV cache reuse)

*본 연구는 2026년 과학기술정보통신부 및 정보통신기획평가원의 AI중심대학 사업 지원을 받아 수행되었음(2026-0-00036)
† Corresponding author

을 지원하는 오픈소스 KV 캐싱 계층으로, 추론 엔진에서 생성된 KV 캐시를 GPU 외부로 추출 · 보관하고 후속 쿼리에 다시 로드해주는 역할을 한다. 다만 그 효과는 얼마나 많은 캐시를 보존하는가뿐 아니라 얼마나 빠르게 다시 로드할 수 있는가에도 좌우된다. 외부 저장 계층의 접근 지연이 크면 KV 캐시 로드 시간이 prefill 연산 시간을 능가하여 재사용 자체의 이점이 상쇄되기 때문이다. 따라서 LMCache가 효과적으로 동작하기 위해서는 보존 용량과 접근 속도를 동시에 만족하는 메모리 계층이 요구된다.

CXL-SSD 메모리 구조와 NAND 미스 지연 문제. CXL의 CXL.mem 프로토콜은 프로세서와 장치를 로드/스토어(Load/Store) 인터페이스로 연결해 캐시라인 단위의 바이트 접근(Byte-addressable)을 지원한다.[3] 이를 활용한 CXL-SSD는 서브 마이크로초(Sub-µs) 지연 시간의 소용량 DRAM 캐시와 테라바이트(TB)급 대용량 NAND 플래시를 결합한 메모리 시맨틱(Memory-semantic) 아키텍처를 갖는다. 호스트 CPU는 이 장치를 호스트 관리형 장치 메모리(HDM)로 인식하여, DRAM 캐시 적중(Hit) 시에는 주 메모리 수준의 빠른 속도를 보인다. 그러나 캐시 미스(Miss) 시에는 백엔드의 낸드 플래시로 요청이 전달되어 수십 마이크로초(Tens-of-µs)의 심각한 지연이 발생하므로, 전체 성능을 극대화하기 위해서는 이 막대한 미스 지연 시간을 효과적으로 은닉(Hiding)하는 최적화가 필수적이다. 따라서 CXL-SSD의 성능을 극대화하기 위해서는 NAND 미스 지연 시간을 효과적으로 은닉하는 캐시 관리 기법이 핵심 과제로 요구된다.

Cylon: CXL-SSD 에뮬레이터. 현재 CXL-SSD의 성능 최적화를 연구하기에는 내부 구조가 불투명한 프로토타입이나 속도가 느린 시뮬레이터 등 기존 실험 환경이 매우 부족한 실정이다. 이를 극복하기 위해 NAND 채널 · 다이 병렬성과 read/program/erase 지연 등 SSD 내부 타이밍을 정밀하게 모사하는 오픈소스 SSD 에뮬레이터 FEMU[4]를 기반으로 한 CXL-SSD 에뮬레이터인 Cylon[2]이 제안되었으며, 본 연구에서는 이를 활용하여 실험을 진행하였다. 기존 FEMU는 모든 메모리 접근에서 VM-exit, 즉 게스트 VM이 호스트 커널로 제어권을 넘기는 과정이 발생하여 수 µs의 오버헤드가 불가피하였다. NVMe SSD 에뮬레이션에서는 이 오버헤드가 문제되지 않았으나, CXL-SSD의 DRAM 캐시 적중 시 요구되는 수백 ns 수준의 응답을 재현하기에는 치명적인 병목으로 작용하였다. Cylon은 하드웨어 가상화의 주소 변환 테이블인 EPT 엔트리를 동적으로 전환하여 캐시 적중 시 VM-exit 없이 DRAM 캐시에 직접 접근하도록 구현하여, 게스트 OS와 애플리케이션의 수정 없이 수백 ns의 캐시 적중과 수십 µs의 캐시 미스 지연 시간을 정확하게 재현할 수 있었다.

3. CXL-SSD 환경 구축 및 LMCache 연동

본 연구에서는 Cylon 에뮬레이터를 활용하여 CXL-SSD 환경을 구축하였다. Cylon은 게스트 VM에 CXL-SSD를 CXL Type-3 메모리 장치로 노출하는 기능을 기본으로 제공하며, 리눅스의 DAX(Direct Access) 드라이버를 통해 블록 I/O 계층을 우회하고 바이트 단위로 직접 접근이 가능한 devdax 장치(/dev/dax0.0)로 인식시킬 수 있다. 이를 통해 게스트 OS에서 CXL-SSD 영역을 일반 메모리처럼 직접 접근할 수 있는 환경을 구성하였다.

LMCache를 CXL-SSD와 연동하기 위해 LMCache에 내장된 DaxBackend 플러그인을 사용하였다. yaml 설정에서 storage_plugins에 dax를 등록하고 dax.device_path를 /dev/dax0.0으로 지정하였으며, store_location과 retrieve_locations를 모두 ”dax”로 설정하여 KV 캐시가 CXL 영역에만 저장 · 로드되도록 하였다. 이때 LMCache의 CPU 메모리는 KV 캐시 캐싱이 아닌 단순 스테이징 버퍼로만 동작한다. 또한 추론 엔진 없이 I/O 경로만 평가하기 위해, Standalone 모드로 LMCacheEngine을 초기화한 뒤 엔진의 store/retrieve API를 직접 호출하였다. 호출은 엔진 내부의 TokenProcessor와 StorageManager를 거쳐 DaxBackend로 전달되므로, 실제 LLM 추론 환경의 KV 캐시 입출력 패턴을 그대로 재현할 수 있다.

4. 실험 및 평가

본 연구에서는 두 가지 주요 실험을 수행하였으며, 실험 환경은 표 1과 같다. 첫 번째 실험에서는 LMCache Standalone 환경에서 호스트 DRAM과 CXL-SSD를 각각 단일 스토리지 백엔드로 구성하였을 때의 KV 캐시 Store/Retrieve 지연 시간 및 대역폭을 비교하여 기본 동작 성능을 평가하였다. 두 번째 실험에서는 KV 캐시의 청크 단위 순차 접근 특성을 고려하여, Cylon의 next-N 프리페치 정책을 활용한 성능 변화를 분석하였다. next-N 프리페치는 캐시 미스 발생 시 요청된 페이지뿐 아니라 그 다음 N개의 연속 페이지를 NAND에서 DRAM 캐시로 함께 끌어오는 방식으로, N(prefetch degree)이 클수록 후속 순차 접근의 미스를 적중으로 전환할 가능성이 커진다. 본 실험에서는 N을 1, 2, 4, 8로 변화시키며 Retrieve 지연 시간 및 대역폭의 변화를 측정하였다.

4.1 Baseline 성능 분석: DRAM vs. CXL-SSD

그림 1은 LMCache Standalone 환경에서 Working Set Size(WSS)를 변화시키며 측정한 DRAM과 CXL-SSD 간의 Store/Retrieve 지연 시간 및 대역폭을 비교한 결과이다. WSS가 CXL-SSD 내부 DRAM 캐시 용량인 4.8GB 이내에 해당하는 캐시 히트 구간에서는 CXL-SSD가 호스트 DRAM과 대등한 수준

표 1: 실험환경

구분	세부사항
CPU	Intel Xeon Gold 5218R @ 2.10 GHz
VM	Ubuntu 22.04, Linux kernel 6.4.6, vCPU 8, DRAM 96 GB
CXL-SSD emulator	Cylon, DRAM cache 4.8 GB, NAND 96 GB, Read 40 µs / Write 200 µs
LMCache 설정	Standalone 모드, 청크 크기 256 tokens (128 KB), WSS 0.25 – 8 GB

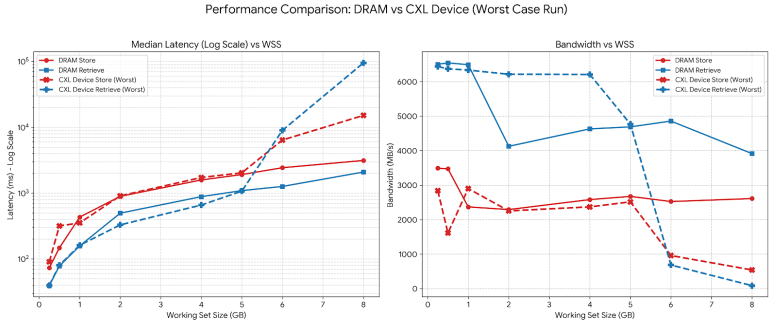

그림 1: DRAM과 CXL 디바이스 간 성능 비교 (Worst-case 환경): Working Set Size 증가에 따른 (좌) 지연시간과 (우) 대역폭 변화

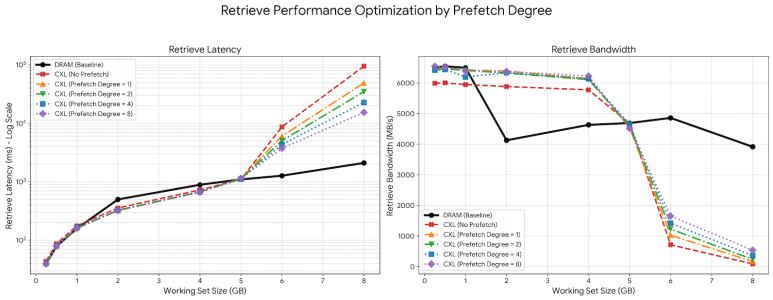

그림 2: 프리페치 정도에 따른 Retrieve 성능 최적화: Working Set Size 증가에 따른 (좌) 지연시간과 (우) 대역폭 변화

의 지연 시간과 대역폭을 기록하였다. 이는 CXL.mem 프로토콜 기반의 바이트 단위 접근이 효과적으로 동작하고 있음을 보여주며, CXL-SSD가 주 메모리 계층에 통합되어 동작할 수 있는 가능성을 실증한다. 그러나 WSS가 약 5GB를 초과하는 시점부터 지연 시간이 급격히 증가하는 양상이 나타났다. 이는 캐시 용량 초과로 인한 미스가 발생하면서 수십 µs에 달하는 NAND Flash 접근 지연이 그대로 반영된 결과로, 8GB 구간에서 CXL Retrieve 지연 시간은 약 100,000ms에 달하며 전체 I/O 효율성을 크게 저해하는 요인으로 작용하였다.

4.2 Prefetch Degree의 변화에 따른 성능 평가

그림 2는 Cylon의 프리페치 기능을 활용하여 Prefetch Degree(No Prefetch, 1, 2, 4, 8)에 따른 Retrieve 지연 시간 및 대역폭 변화를 측정한 결과이다. WSS가 DRAM 캐시 용량 이내인 캐시 히트 구간에서는 Prefetch 설정 유무와 관계없이 약 6,000 MB/s 이상의 최고 수준 읽기 성능을 유지하여, 히트 구간에서는 prefetch가 추가적인 이득 없이 정상 동작함을 확인하였다. 반면 WSS가 6GB 8GB 구간에 진입하자 설정별 성능 격차가 극명하게 드러났다. No Prefetch 환경에서는 빈번한 캐시 미스와 NAND Flash 접근이 중첩되면서 8GB 기준 지연 시간이 약 94,000ms까지 치솟고 대역폭은 86 MB/s로 급락하는 심각한 병목 현상이 발생하였다. 반면 Prefetch Degree를 점진적으로 높일수록 지연시간 증가폭이 단계적으로 억제되었으며, Degree 8 환경에서는 동일 구간의 지연 시간이 약 15,321ms, 대역폭은 534 MB/s 수준으로 대폭 개선되었다. 이는 높은 Degree의 프리페치가 다수의 지연된 I/O 요청을 효과적으로 병합·중첩 처리함으로써, CXL-SSD 특유의 용량 한계 초과 시 발생하는 성능 붕괴 현상을 완화하고 스토리지 접근 효율을 극대화할 수 있음을 입증한다.

5. 결론 및 향후 연구

본 연구에서는 CXL-SSD를 LMCache의 확장 메모리 백엔드로 활용할 때의 효용성과 한계를 실험적으로 확인하였다. CXL-SSD는 내부 DRAM 캐시 적중 구간에서 호스트 DRAM과 대등한 성능을 보여 메모리 확장 계층으로서의 가능성을 입증하였다. 반면, 워크로드가 캐시 용량을 초과하는 미스 구간에서는 NAND Flash 접근 지연으로 인해 성능이 크게 저하되었으며, 스토리지 내부 프리페치 기법을 통해 이를 부분적으로 완화할 수 있음을 확인하였다. 향후에는 호스트 및 애플리케이션 수준에서 I/O 지연을 선제적으로 은닉하기 위한 소프트웨어 최적화 연구를 진행할 계획이다.

참고 문헌

[1] Y. Liu, Y. Cheng, J. Yao, Y. An, X. Chen, S. Feng, Y. Huang, S. Shen, R. Zhang, K. Du, and J. Jiang, "Lmcache: An efficient kv cache layer for enterprise-scale llm inference," 2025.

[2] D. Yoon, H. Idden, J. Liu, B. Inceisci, S. H. Noh, and H. Li, "Cylon: Fast and accurate Full-System emulation of CXL-SSDs," in *24th USENIX Conference on File and Storage Technologies (FAST 26)*, (Santa Clara, CA), pp. 313–327, USENIX Association, Feb. 2026.

[3] Compute Express Link, "Compute express link homepage." https://www.computeexpresslink.org, 2026.

[4] H. Li, M. Hao, M. H. Tong, S. Sundararaman, M. Bjørling, and H. S. Gunawi, "The CASE of FEMU: Cheap, accurate, scalable and extensible flash emulator," in *16th USENIX Conference on File and Storage Technologies (FAST 18)*, (Oakland, CA), pp. 83–90, USENIX Association, Feb. 2018.