OS 세계에서는 AI의 발전에 따른 병목을 어떻게 대응하고 있는가?

논문으로 본 AI-OS Co-design 흐름과 쟁점들

한국외국어대학교 컴퓨터공학부 한정연

패러다임의 전환: Disk I/O에서 메모리 병목으로

과거: Computer Vision (CV) 시대

핵심 병목: Disk I/O (입출력 속도)

수만 장의 대용량 이미지 파일을 스토리지(디스크)로부터 읽어오는 속도가 주요 병목 요인이었습니다.

OS 커널의 주요 과제는 파일 시스템 입출력 최적화, 스토리지 캐싱, 효율적 프리페칭(Prefetching) 구조 설계였습니다.

현재: LLM (대규모 언어 모델) 시대

핵심 병목: Memory Capacity & Bandwidth

완전히 새로운 양상의 메모리 문제가 발생하였으며, 핵심 원인은 바로 **KV Cache** 입니다.

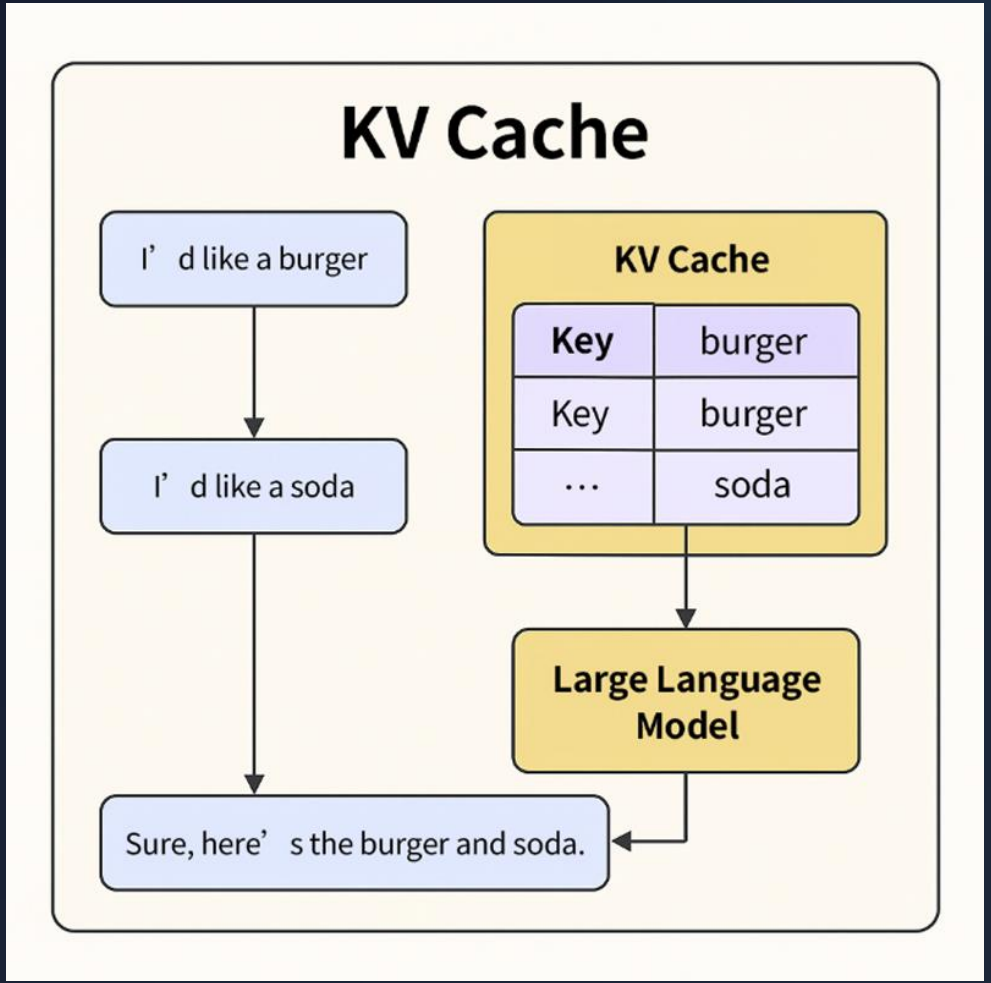
매 단어 생성 시 거대한 임시 기억 데이터를 연속 유지해야 하며, 이 데이터가 커널의 기존 메모리 관리 방식과 정면 충돌합니다.

LLM의 필수 기억 장치: KV Cache

KV Cache (Key-Value Cache)란?

LLM이 문장을 생성할 때 이전 단어들의 연산 결과를 버리지 않고 메모리에 임시로 쌓아두는 '기억 장치'입니다.

- **연산 재사용:** 매번 처음부터 다시 계산하지 않고 이전 기억을 계속 재활용합니다.
- **메모리 급증:** 대화 문맥(Context)이 길어질수록 데이터 크기가 수 GB 단위로 폭발적으로 늘어납니다.
- **커널과의 마찰:** 거대하고 불규칙한 데이터를 기존 커널 방식으로 관리할 때 심각한 효율 저하가 발생합니다.



메모리 파편화(Fragmentation)와 주차장 비유

🅿 주차장 비유로 보는 파편화

전체 빈자리를 합치면 **100대 분량(100GB)**의 잔여 공간이 존재하지만, 차들이 들락날락하며 1~2대씩 흩어져 있는 상태입니다.
이때 50대 연속으로 주차해야 하는 **대형 버스(50GB KV Cache)**가 진입하면, 전체 용량은 넉넉함에도 연속 공간이 없어 주차가 거절됩니다.

⚠ 커널의 4KB 단위와 OOM 에러

기존 리눅스 커널의 기본 메모리 할당 단위는 **4KB** 입니다.

LLM이 GB 단위의 거대한 연속 메모리를 요청할 때 커널이 빈 4KB 조각들을 모아 연속 블록을 만들어내지 못하면, **메모리가 남아있어도 OOM(Out-of-Memory) 에러로 시스템이 멈춥니다.**

KV Cache가 메모리 파편화에 취약한 이유

1. 동적 크기 증가

사용자와 대화를 주고받을 때 토큰(단어)이 하나 생성될 때마다 KV Cache 데이터의 크기가 실시간으로 가변적으로 늘어납니다.

2. 예측 불가능한 길이

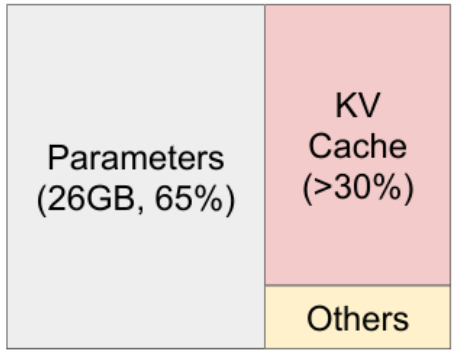
사용자의 질문이 짧은 단답형일지, 긴 문서 작성일지 런타임 사전 시점에서는 정확한 길이를 인지하는 것이 불가능합니다.

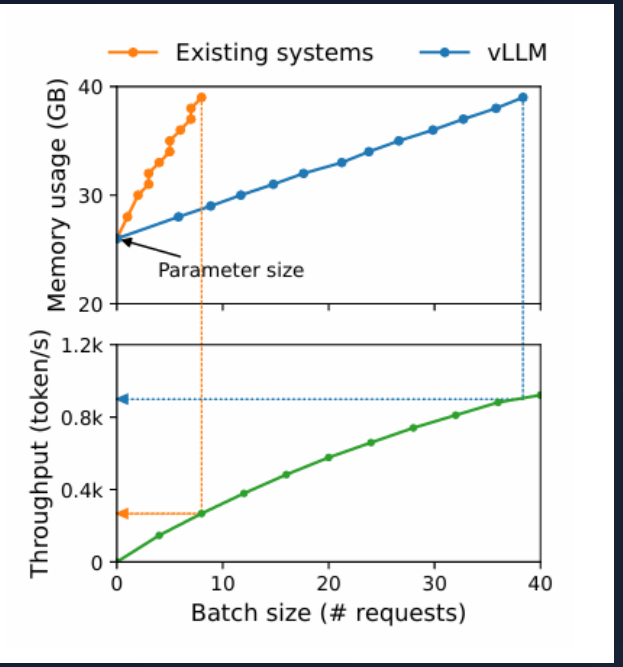
3. 메모리 이주 및 복사 오버헤드

연속 공간이 부족해지면 기존 KV Cache를 새로운 대용량 연속 버퍼로 데이터 복사하거나, 커널 단의 메모리 컴팩션을 수행해야 하므로 심각한 지연시간이 발생합니다.

vLLM의 혁신: PagedAttention을 통한 우회

Efficient Memory Management for Large Language Model Serving with PagedAttention
SOSP 2023 (ACM Symposium on Operating Systems Principles)





vLLM PagedAttention 방식

KV Cache를 16개 토큰 크기의 **작은 '블록(Block)' 단위**로 잘게 쪼개어 빈 곳에 흩뿌려 저장합니다.

Block Table (주소록)을 생성해 물리 위치를 가상 매핑함으로써 **'연속 공간 제약' 자체를 제거**했습니다.

유저 공간 페이징에서 커널 통합으로

GMem: Efficient OS Support for Low-Latency GPU Memory Management

ASPLOS 2024 (Architectural Support for Programming Languages and Operating Systems)

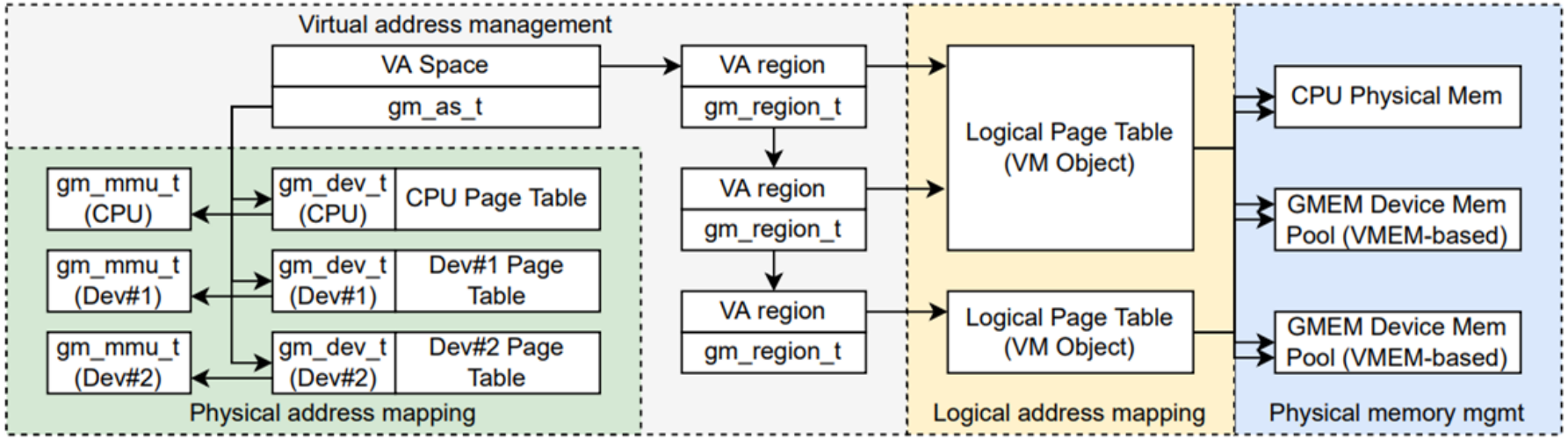


Figure 1: GMEM re-architects the core VM system for peripheral MMUs.

Trend 2: 커널 내부 통합 (GMem)

OSDI '24 GMem 연구처럼, 임시방편을 멈추고 리눅스 커널 VMM이 가속기 메모리를 직접 통제하는 커널 통합 흐름이 형성되었습니다.

CXL 이종 메모리(Tiered Memory)의 등장

Pond: CXL-Based Memory Pooling Systems for Cloud Platforms
ASPLOS 2023 (Architectural Support for Programming Languages and Operating Systems)

HBM의 한계와 CXL 확장

HBM(고대역폭 메모리)의 비싼 가격과 물리적 용량 한계를 극복하기 위해 PCIe 고속 버스로 테라바이트급 CXL DRAM을 연결하는 이종 메모리가 필수화되었습니다.

Semantic Gap (정보 격차) 발생

CXL 메모리는 용량이 크지만 PCIe 버스를 거치므로 느립니다. 기존 커널 방식처럼 단순 과거 접근 기록(LRU)만 보면, 당장 연산할 Hot 데이터가 느린 CXL에 방치되어 **지연시간 폭탄**을 맞게 됩니다.

AI-Aware Kernel Interface 과제

LLM의 사전 결정론적(Deterministic) 특성 LLM은 다음 단어를 만들 때 어떤 메모리를 읽을지 수학적으로 100% 사전에 결정되어 있는 특성을 가집니다.

AI 런타임의 힌트 제공 AI 런타임이 미래의 메모리 접근 패턴을 미리 계산하여 OS 커널에 힌트(Hint) 형태로 전달합니다.

CXL 버스 병목 없는 Seamless Migration 커널은 전달받은 힌트를 바탕으로 CXL과 HBM 간 데이터 교대를 버스 병목 없이 매끄럽게 선제적으로 수행합니다.

AI-OS Co-design의 3대 쟁점

“ 학계의 Co-design 아이디어들이 리눅스 메인라인(Mainline) 커널에 안착하기 위한 3대 핵심 과제

QUESTION 01

메인라인 커널의 범용성 확보

보편적 OS(General-Purpose)의 아키텍처와 안정성을 파괴하지 않으면서 AI 특화 요구사항을 수용할 커널 인터페이스를 어떻게 설계할 것인가?

QUESTION 02

AI-Aware 예측형 메모리 관리

CXL 지연시간 병목을 극복하기 위해 AI 런타임의 사전 결정론적 힌트를 수용하는 예측형 메모리 제어 계층을 어떻게 구체화할 것인가?

QUESTION 03

HW 오프로딩과 표준화 장벽

PIM 및 CXL 컨트롤러 내부 연산 시 OS의 제어권(Control)과 하드웨어 오프로딩 사이의 표준화 장벽을 어떻게 넘어설 것인가?

감사합니다

논문으로 본 AI-OS Co-design 흐름과 쟁점들

한국외국어대학교 컴퓨터공학부 한정연