MODULE 08 · 심화

추론 인프라의 최전선 — 프리필·디코드 분리

모듈 2에서 "AI의 대답은 읽기(프리필)와 쓰기(디코드), 두 단계"라고 배웠습니다. 지난 1년 사이 세계 AI 인프라 업계는 이 두 단계를 **아예 다른 컴퓨터에 나눠서 돌리는 방식**으로 조용히 갈아탔습니다. 왜 그랬는지, 그게 어떤 결과를 냈는지, 그리고 소규모 장비(GPU 한두 장)인 우리에게 무슨 의미인지 — 배경지식 없이도 따라올 수 있게 풀어봅니다. (2026년 중반 기준, 빠르게 변하는 분야입니다.)

01 어느 SNS 포스트의 도발 — "당신은 아직 옛날 방식으로 계산한다"

이 모듈의 출발점은 한 국내 엔지니어의 SNS 포스트입니다. 요지는 도발적입니다 — "AI 추론 인프라 시장은 지난해 패러다임이 근본적으로 바뀌었는데, 대부분은 여전히 **GPU 용량 계산(모델 무게 + KV 캐시)만** 하고 있다"는 것. 그러면서 이런 수수께끼를 던집니다:

> 💡 **비유**
>
> "AI 서비스를 하려면 서버가 10배 더 필요하다"는 말과 "토큰(AI가 만드는 글자) 비용은 10배 싸졌다"는 말이 **동시에 사실**입니다. 어떻게 그럴 수 있을까요? — 이 모듈을 다 읽으면 이 모순이 풀립니다.

참고로 "모델 무게 + KV 캐시"로 GPU 용량을 계산하는 것은 정확히 우리가 모듈 3~4에서 배운 산정법입니다. 그 산정법이 틀렸다는 게 아닙니다 — **한 대의 컴퓨터 안에서는 여전히 그게 정답**입니다. 바뀐 것은 수십~수백 대의 GPU로 대규모 서비스를 할 때의 설계 방식입니다.

02 복습 — 읽기와 쓰기는 성격이 정반대인 일이다

모듈 2의 핵심을 한 번 더 짚습니다. AI가 답을 만드는 과정은 두 단계입니다:

- **프리필(읽기)** — 질문과 문서를 한꺼번에 읽어들이는 단계. 계산이 몰아쳐서 **연산 장치가 바쁩니다**. 긴 문서 하나만 읽어도 GPU가 꽉 찹니다.
- **디코드(쓰기)** — 답을 한 글자씩 만들어내는 단계. 매 글자마다 모델 전체를 **메모리에서 다시 읽어와야** 해서, 연산보다 **메모리 왕복이 병목**입니다. 혼자서는 GPU가 놀아서, 수십 명의 요청을 겹쳐 돌려야 효율이 납니다.

> 💡 **비유**
>
> 식당으로 비유하면 — 프리필은 **주방의 화력**이 필요한 일(요리 하나에 불을 다 씀)이고, 디코드는 **홀 서빙**(한 번 나갈 때 여러 테이블 것을 같이 나르면 효율적)입니다. 성격이 정반대인 두 일을 한 사람에게 동시에 시키면, 요리하다 서빙 놓치고 서빙하다 불 태웁니다.

지금까지는 이 정반대인 두 일을 **같은 GPU 한 장**이 번갈아 했습니다. 그래서 어쩔 수 없는 손해가 생깁니다 — 긴 문서를 읽는 동안(프리필) 답을 기다리던 다른 사용자들의 쓰기(디코드)가 멈추고, 반대로 쓰기를 겹쳐 돌리자니 읽기가 끼어들 자리가 없습니다.

03 해법 — 주방과 홀을 아예 분리한다 (PD 분리)

업계가 찾은 답은 단순하고 과감합니다. **프리필 전담 GPU 무리와 디코드 전담 GPU 무리를 아예 따로 두는 것**입니다. 영어로 PD 분리(Prefill-Decode Disaggregation)라고 부릅니다. 읽기가 끝난 중간 결과(KV 캐시 — 모델이 읽은 내용의 메모장)를 디코드 무리로 넘겨주면 됩니다.

실제 규모의 사례로, 대형 오픈 모델 DeepSeek-V3의 운영 구성은 **프리필 32장 : 디코드 320장** — 대략 1:10입니다. 이렇게 나눴더니:

- 같은 GPU 수로 처리량(동시에 소화하는 요청량)이 **약 2.5배**,
- 수용 용량은 **약 5배(498%)** 늘었고,
- 토큰 백만 개 생산 원가가 **약 $0.20** 수준까지 떨어졌다는 실측이 공개됐습니다(같은 급 API 판매가는 ~$1.00).
- 또 다른 실측: 통합 방식에선 대형 모델이 사실상 1명분 서비스밖에 못 하던 H200 8장 구성이, 분리 후 **동시 175명+ 배치**를 소화.

이제 첫 수수께끼가 풀립니다. **서버가 10배 필요해진 것**은 프리필·디코드를 나누고 디코드 쪽에 GPU를 몰아주는 새 설계 때문이고(1:10을 보세요), **토큰 값이 10배 싸진 것**은 그렇게 나눈 덕에 GPU 한 장이 노는 시간 없이 일해서입니다. 장비는 늘고, 장비당 생산성은 더 크게 늘었습니다 — 그래서 GPU 판매량이 폭증하면서 동시에 토큰 값이 떨어지는, 언뜻 모순 같은 두 곡선이 함께 갑니다.

04 MoE와 만나면 — 악순환이 선순환으로

모듈 3에서 배운 MoE(전문가 혼합 — 필요한 전문가만 깨워 쓰는 모델)와 PD 분리는 특히 궁합이 좋습니다. 통합 방식의 MoE 서빙에는 악순환이 있었습니다:

- **악순환(통합)**: KV 캐시가 메모리를 잡아먹음 → 배치(동시 요청 수)를 못 키움 → 전문가들이 놀게 됨 → 효율 급락.
- **선순환(분리)**: 디코드 GPU들이 전문가를 나눠 맡음 → 장당 모델 부담 감소 → 남은 메모리로 배치 확대 → 전문가들이 쉴 틈 없이 일함.

대형 MoE 모델 Kimi K2.7이 H200 128장에서 초당 디코드 토큰 28만 8천 개를 낸 실측이 이 조합의 대표 사례입니다. 우리가 주력으로 고른 Gemma 4 26B-A4B도 MoE죠 — 규모는 다르지만, "MoE는 배치를 채워야 진가가 나온다"는 원리는 우리 서버의 `--parallel` 슬롯 설정에도 그대로 적용됩니다.

05 도구 이름표 정리 — 뉴스에서 마주칠 그 이름들

이 분야 글에는 도구 이름이 쏟아집니다. 각각 무슨 역할인지 이름표만 붙여두면 뉴스가 읽힙니다:

이름	역할 (비유)
vLLM / SGLang	대규모 서빙 엔진 — 식당의 주방 시스템 전체. 우리 llama.cpp의 "수백 GPU급 형님들"(모듈 4)
Dynamo (NVIDIA)	분리된 주방·홀 전체를 지휘하는 총괄 매니저(오케스트레이터)
Mooncake	읽어둔 내용(KV 캐시)을 GPU 밖 일반 메모리·SSD까지 넓혀 보관하는 창고 시스템
LMCache	같은 문서를 또 읽을 때 재활용하도록 캐시를 나눠주는 배급소

하드웨어도 이 방향으로 진화 중입니다 — NVIDIA의 차세대 플랫폼은 프리필용 칩과 디코드용 칩을 **아예 따로 설계해 한 시스템에 담는** 구성을 예고했고, 모듈 7에서 본 텐스토렌트는 초고속 내장 메모리(SRAM)로 디코드 특화 장비를 훨씬 싼 가격에 내놓는 전략입니다. "읽기와 쓰기는 다른 일"이라는 인식이 칩 설계까지 내려간 겁니다.

06 그래서 우리에게는? — 원리는 같고, 규모만 다르다

솔직하게: PD 분리는 **수십 장 이상의 GPU로 대규모 서비스를 할 때의 설계**입니다. GPU 한두 장인 우리 서버에 가져올 기술은 아닙니다. 그런데 이 모듈을 심화에 넣은 이유가 있습니다:

- **우리 벤치가 정확히 이 감각을 기릅니다.** 벤치 기록지에 프리필 속도와 디코드 속도를 따로 적게 되어 있는 것, 기억하나요? 세계 최전선 인프라가 두 단계를 물리적으로 분리하는 이유와, 우리가 두 숫자를 따로 재는 이유는 같은 통찰입니다 — **읽기와 쓰기는 다른 일이다.**
- **산정의 언어가 확장됩니다.** "모델 무게 + KV 캐시"는 한 대 안의 산정법이고, 그 위에 "프리필:디코드 비율"이라는 함대 단위 산정법이 얹힌 것 — 우리 판단 프레임(메모리→모델→엔진→실측)이 낡은 게 아니라, 그 위층이 생긴 겁니다.
- **도입 논의에서 헛돈 쓰지 않게 됩니다.** 언젠가 기관이 대규모 추론 인프라를 검토할 때, "GPU 몇 장"이 아니라 "프리필·디코드를 어떻게 나눌 설계인가"를 물을 수 있는 사람이 됩니다.

우리 연구회에선

이 모듈의 출처: Threads @slamslam__ 포스트(문제 제기)와 CV-Learn "효율적 AI 인프라"(기술 해설·실측 수치). 인용한 수치(2.5배·498%·$0.20/M 등)는 해당 글이 정리한 업계 공개 실측이며, 독립 검증된 학술 수치는 아닙니다 — 방향을 읽는 자료로 보세요.

↗ 심화 학습

더 깊이: 위 CV-Learn 원문이 DeepSeek·Kimi·Mooncake의 공개 자료를 잘 엮어둔 한국어 해설입니다. 영어가 괜찮다면 vLLM·SGLang 공식 블로그의 disaggregation 글들이 1차 소스입니다. · 기술 매체·블로그 성격의 자료이니 수치는 "그 팀의 실측 보고" 수준으로 받아들이세요.

07 퀴즈 — 인접 개념을 가려내기

CHECK

이해했는지 확인해 봅시다

틀려도 괜찮습니다 — 오답을 고르면 왜 아닌지 설명이 나오고, 정답을 찾을 때까지 다시 고를 수 있습니다. 점수는 첫 시도 기준입니다.

문제 1 / 5

Q1. 프리필과 디코드를 서로 다른 GPU 무리에 나눠 돌리는 근본 이유는?

① 두 단계가 쓰는 모델 파일이 서로 달라서

② 프리필은 연산이, 디코드는 메모리 왕복이 병목 — 자원 요구가 정반대라서

③ 프리필이 끝난 요청을 더 싼 GPU로 옮겨 전기료를 아끼려고

④ 보안상 읽기와 쓰기를 같은 장비에서 하면 안 되기 때문에

← 이전
모듈 7 · AI 반도체 생태계

다음 →
모듈 9 · 분산 클라우드