MODULE 01

내 장비에서 어떤 모델까지 돌릴 수 있나

로컬 AI를 시작하기 전에 가장 많이 받는 질문이 있습니다. "내 사무실 PC로 뭘 돌릴 수 있나요?" 솔직한 답은 "장비를 봐야 안다"입니다 — 이 모듈을 읽고 나면 어떤 장비를 보든 그 답을 스스로 낼 수 있게 됩니다.

01 CPU와 GPU — 일꾼의 종류가 다르다

컴퓨터 안에는 계산을 담당하는 두 종류의 부품이 있습니다. **CPU**는 소수의, 그러나 매우 똑똑한 일꾼입니다. 복잡한 판단을 내리거나 순서대로 처리해야 하는 작업에 강합니다. 반면 **GPU**는 수천 개의 단순한 일꾼입니다(이 일꾼 하나하나를 '코어'라고 부릅니다). 한 명 한 명은 어려운 일을 못 하지만, 똑같은 계산을 동시에 아주 많이 처리할 수 있습니다.

그런데 LLM(거대언어모델)이 추론할 때 실제로 하는 일은 **거대한 행렬 곱셈**(행렬=숫자를 격자 모양 표로 늘어놓은 것)입니다. 곱하고 더하는 단순한 계산을 어마어마한 양으로 반복하는 것이 전부입니다. 이런 성격의 작업에서는 GPU가 압도적으로 유리합니다.

> 💡 **비유**
>
> 세금 신고서를 작성하는 일과, 봉투 1만 장에 우표를 붙이는 일을 비교해 보세요. 세금 신고는 복잡한 판단이 필요해서 전문가 한 명(CPU)이 맡는 게 맞습니다. 하지만 우표 붙이기는 단순한 동작의 반복이라, 아르바이트생 1000명(GPU)을 동시에 투입하는 쪽이 압도적으로 빠릅니다.

그리고 CPU도 LLM을 돌릴 수는 있습니다. 작은 모델(1B~4B급)이라면 GPU 없이 CPU만으로도 돌아갑니다 — 느리지만, 개념을 익히는 실습에는 충분합니다. 이 지점이 우리에게 중요한데, 아래 04에서 다시 다룹니다.

02 RAM과 VRAM — 책상이 두 개다

계산을 하려면 데이터를 올려둘 작업 공간이 필요합니다. CPU에게는 **RAM**이 그 책상이고, GPU에게는 **VRAM**이라는 별도의 책상이 있습니다. VRAM은 그래픽카드에 직접 붙어 있는, GPU 전용 메모리입니다.

모델이 제 속도로 돌아가려면 모델 전체가 이 VRAM 위에 올라가 있어야 합니다. 만약 모델 크기가 VRAM 용량보다 크면, 넘치는 부분은 RAM과 CPU 쪽으로 떠넘겨집니다. 이를 **오프로딩**이라고 부릅니다. 오프로딩은 됩니다. 다만 느립니다 — 속도가 수 배에서 수십 배까지 느려집니다. 즉 "돌아간다"는 것과 "업무에 쓸 만하다"는 것은 전혀 다른 이야기입니다.

- ✔ **VRAM 용량이 곧 상한선입니다** — 내 GPU에 올릴 수 있는 모델 크기의 한계를 정합니다.
- ✔ **오프로딩은 최후의 수단입니다** — 급할 때 쓸 수는 있지만, 이것을 전제로 장비를 고르면 안 됩니다.
- ✔ **그래서 GPU 스펙을 볼 때 가장 먼저 확인할 숫자는 VRAM GB입니다.**

03 속도를 정하는 진짜 범인 — 메모리 대역폭

다소 의외인 사실 하나를 말씀드리겠습니다. 모델이 토큰(글자 조각) 하나를 만들어낼 때마다, GPU는 **모델 전체의 무게(파라미터)를 메모리에서 처음부터 끝까지 한 번 다시 읽습니다.** 글자 하나를 쓸 때마다 모델 통째로 훑는 셈입니다.

그렇다면 생성 속도의 이론상 한계는 결국 "메모리를 얼마나 빨리 읽을 수 있는가"에 달려 있습니다. 이를 식으로 쓰면 대략 이렇습니다.

> **생성 속도의 이론상 상한 ≈ 메모리 대역폭(GB/s) ÷ 모델 크기(GB)**
> 예를 들어 초당 700GB를 읽을 수 있는 GPU에 14GB짜리 모델을 올리면, 이론상 최대 초당 50토큰까지 나올 수 있습니다. 다만 실제로는 여러 병목 때문에 이론값의 절반 이하에 그치는 경우가 흔합니다.

여기서 얻는 실용적인 시사점은 이렇습니다. 같은 GPU를 쓴다면, 모델 파일이 작을수록 더 빠릅니다. 그래서 모델 크기를 줄이는 기술이 중요한데, 이 방법은 모듈 3(양자화)에서 자세히 다룹니다.

대역폭은 한 종류가 아니다 — GPU 안의 길과 GPU 사이의 길

위에서 말한 대역폭은 **GPU와 자기 메모리(VRAM) 사이**의 속도입니다. 그런데 길이 하나 더 있습니다 — **GPU와 GPU 사이, GPU와 CPU 사이**를 잇는 연결 통로입니다.

> 💡 **비유**
>
> 도서관 사서(GPU)와 그 사서의 책상(VRAM) 사이는 손만 뻗으면 되는 초고속입니다(HBM 같은 고대역폭 메모리라면 더욱). 그런데 옆방 사서에게 책을 넘기려면 **복도**를 지나야 합니다. 이 복도가 PCIe인데, 책상 속도의 수십 분의 일밖에 안 됩니다. NVLink는 이 문제를 풀려고 NVIDIA가 사서들 사이에 깔아준 **전용 고속 컨베이어 벨트**입니다 — 최신 세대 기준 PCIe보다 수십 배 빠릅니다.

- ✔ **모델이 GPU 한 장에 다 들어가면** 복도는 거의 쓸 일이 없습니다 — VRAM 대역폭만 중요합니다.
- ✔ **모델을 여러 장에 나누면** 장과 장 사이로 중간 결과가 오갑니다 — 이때 복도(PCIe/NVLink)가 병목이 될 수 있습니다.
- ✔ H100·H200급 장비가 비싼데도 팔리는 이유가 여기 있습니다: 연산력·HBM에 더해 **NVLink로 GPU 여러 장을 "사실상 한 장처럼" 묶는 능력**이 초대형 모델 시대의 핵심이라서입니다.

> 🔗 **우리 연구회에선**
>
> 우리 서버가 "구형"인 이유도 정확히 이 지점입니다. GPU 두 장이 구세대 복도(PCIe 3.0)로만 연결되어 있어서, 장 사이 통신이 많은 분할 방식은 불리합니다(예: 층을 안 나누고 계산 자체를 쪼개는 다른 분할 방식도 있는데, 이런 방식은 장 사이에 훨씬 자주 데이터를 주고받아야 합니다). 그래서 우리는 통신량이 적은 **레이어 분할**(모델을 층별로 잘라 각 장이 자기 층만 처리하고 바통을 넘기는 방식)을 씁니다 — 제약을 알면 그 제약에 맞는 설계를 고를 수 있다는 좋은 예시입니다.

04 장비 읽는 법 — 확인할 것 세 가지만 보면 된다

이제 실제로 장비 스펙표를 볼 때 무엇을 확인해야 하는지 정리해 보겠습니다. 핵심은 **GPU 이름(세대), VRAM 용량, 메모리 대역폭** 이 세 가지입니다. 스펙표에 함께 적힌 쿠다 코어 수(일꾼 수)나 부스트 클럭(일꾼의 손 빠르기) 같은 숫자들은 속도에 관여하지만, "어떤 모델까지 올라가나"는 결국 VRAM이 정합니다. 참고로 지금 GPU에 VRAM이 얼마나 쓰이고 있는지는 `nvidia-smi` 명령으로 바로 확인할 수 있습니다(검은 화면의 명령창인 **터미널**에 입력하는 전문가용 확인법입니다 — 지금은 몰라도 됩니다. 워밍업 페이지에서 비유와 함께 설명합니다).

장비 예시	VRAM	감각

사무용 PC (GPU 없음, CPU만)	—	1B~4B급 Q4를 CPU로 구동 — 느리지만(초당 수 토큰) 개념 실습 가능
게이밍 노트북 RTX 3070	8GB	4B급 Q4 모델 쾌적
게이밍 노트북 RTX 4080	12GB	8B급 Q4 쾌적, 14B급 도전 가능
서버용 구형 GPU 2장(예: 12GB×2)	합 24GB	20~30B급 양자화 모델을 나눠서 탑재 가능

※ 표의 "Q4"는 모델을 4비트로 압축(양자화)했다는 표기입니다 — 모듈 3에서 자세히 배웁니다. 지금은 "압축해서 가볍게 만든 버전" 정도로 읽으세요.

한 가지 더 유의할 점은, GPU 세대가 너무 낮으면 최신 소프트웨어(예: vLLM 같은 서빙 엔진)가 아예 지원을 끊어버리는 경우가 있다는 것입니다. 이 이야기는 모듈 4(서빙 엔진)에서 다시 다룹니다.

우리 연구회에선

우리 연구회의 현실을 그대로 대입해 봅시다. 회원 대부분의 사무실 PC에는 GPU가 없습니다. 그래서 실습 경로는 세 갈래입니다 — ① **공유 GPU 서버**(구형이지만 대역폭 좋은 서버 GPU 2장, 합 24GB)를 **API**로 쓰는 것이 기본 경로(주소를 하나 받아서 다른 프로그램이 그 서버에 대신 요청을 보내게 하는 방식입니다 — 실제 실습 방법은 별도 가이드로 안내됩니다. 워밍업 페이지에서 비유와 함께 설명합니다), ② 각자 PC에서는 CPU로 1B~4B급 소형 모델을 직접 돌려보며 개념을 익히고, ③ 필요 시 클라우드 GPU를 시간 단위로 빌려 비교 실험. 위 표의 게이밍 노트북 행은 "집에 그런 장비가 있다면"의 참고용입니다.

05 GPU만 있는 게 아니다 — NPU와 AI 반도체 지형

뉴스에서 **NPU**(Neural Processing Unit)라는 말을 자주 접하셨을 겁니다. 좁게는 스마트폰·노트북에 들어가는 AI 보조 칩을, 넓게는 **AI 계산에 특화해 새로 설계된 반도체 전반**을 가리키는 말로 쓰입니다.

지금의 AI 인프라 시장은 NVIDIA GPU의 사실상 독주 상태입니다. 그런데 그 이유가 칩 성능 때문만은 아닙니다. 지난 20년 가까이 쌓인 **CUDA라는 소프트웨어 생태계** — 개발 도구, 라이브러리, 그리고 그것에 익숙한 전 세계 개발자들 — 가 진짜 성벽입니다. 칩을 잘 만들어도 소프트웨어가 따라오지 않으면 아무도 쓰지 않으니까요.

용어 짚기 — CUDA와 드라이버

CUDA는 NVIDIA GPU에게 일반 계산을 시키기 위한 프로그래밍 플랫폼(개발 도구 + 실행 환경)입니다. GPU에서 도는 거의 모든 AI 소프트웨어가 CUDA 위에서 만들어졌습니다. **드라이버**는 운영체제와 GPU를 잇는 기본 소프트웨어인데, 드라이버 버전이 지원하는 CUDA 버전에 상한이 있습니다. 그래서 "드라이버가 오래된 서버에서는 최신 CUDA를 요구하는 도구가 안 도는" 일이 실제로 생깁니다 — 장비 조사에서 GPU 이름과 함께 드라이버·CUDA 버전을 확인하는 이유입니다.

이 독주를 깨려는 도전이 곳곳에서 진행 중입니다. 해외에서는 전설적인 칩 설계자 짐 켈러가 이끄는 **텐스토렌트(Tenstorrent)** 같은 회사가, 국내에서는 **리벨리온·퓨리오사AI** 같은 K-NPU 기업들이 정부 지원과 함께 추론 시장을 공략하고 있습니다. 왜 하필 추론인지, 각 도전자의 전략과 현재 수준은 어떤지 — 이 지형도는 심화 모듈(모듈 7)에서 제대로 다룹니다.

여기서는 실무 감각 하나만 챙겨두시면 됩니다. 제품 광고의 "NPU 탑재" 문구와 "이 장비에서 LLM이 잘 돈다"는 것은 별개의 이야기입니다. 판단 기준은 언제나 **내가 쓸 모델과 서빙 소프트웨어가 그 칩에서 실제로 돌아가는가**입니다.

↗ 심화 학습

정밀 VRAM 산정 도구(권장): **APX ML VRAM Calculator** — 정밀도(Precision)·동시 처리(Batch)·컨텍스트 길이(Seq Len)까지 반영한 실전용 계산기(세 용어의 자세한 뜻은 모듈 2·3에서 다룹니다 — 지금은 계산기가 있다는 것만 기억하세요). 이 세 가지를 무시하고 "VRAM 용량만 맞으면 되지"라고 계산하면 실제 서빙에서 성능이 무너집니다.
가볍게 감 잡기용(부가): **whatcani.run** — 스펙 입력하면 구동 가능 모델 목록을 보여주는 간이 도구.

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 / 4

Q1. LLM 추론에 CPU보다 GPU가 유리한 이유는?

① GPU가 CPU보다 클럭(개별 계산 속도)이 높아서

② LLM 추론이 단순 계산의 대량 병렬 반복이라, 수천 개 코어로 동시에 처리하는 GPU 구조에 맞아서

③ GPU에는 VRAM이 있어 더 많은 데이터를 저장할 수 있어서

④ GPU가 AI 전용으로 설계된 칩이라서

← 이전
모듈 0 · 오리엔테이션

다음 →
모듈 2 · LLM 동작 원리