ISSUE

AI 인프라 경쟁에서 소프트웨어의 구조적 역할

The Structural Role of Software in AI Infrastructure Competition

- **강호준** 소프트웨어정책연구소 AI정책연구실 선임연구원 | hjk_rep@spri.kr
- **안성원** 소프트웨어정책연구소 AI정책연구실 책임연구원 | swahn@spri.kr

EXECUTIVE SUMMARY

2026년 전 세계 AI 지출은 2.5조 달러에 이를 전망이며, 그중 절반 이상이 서버·가속기·데이터센터 등 인프라에 집중된다. 이 투자의 중심에 NVIDIA GPU가 있으며, 데이터센터 GPU 매출의 약 86%를 차지하는 압도적 지배력을 유지하고 있다. 그러나 이러한 지배력은 단순히 하드웨어(HW) 성능에서 비롯된 것이 아니다. 동일한 H100 칩에서도 소프트웨어(SW) 최적화 수준에 따라 실제 처리량이 3배 이상 차이가 발생하며, NVIDIA가 2006년 CUDA 출시 이후 약 20년간 축적한 SW 생태계가 구조적 진입장벽을 형성하고 있다.

보고서는 AI SW 스택을 프레임워크, 컴파일러, 가속 라이브러리, 드라이버/런타임 4계층으로 구분하고, 각 계층이 HW 종속을 형성하는 기술적 경로를 추적한다. 이를 토대로 세 유형의 종속 메커니즘을 도출한다. CUDA 경로에서만 최적 성능이 발휘되는 '성능 종속', JAX-XLA-TPU처럼 SW 선택이 곧 HW를 확정하는 '설계 종속', 폐쇄적 드라이버 구조가 HW 대체를 물리적으로 차단하는 '구조적 종속'이 각각 상이한 메커니즘으로 작동하며, 이 세 유형이 중첩될 때 전환 비용은 기하급수적으로 증가한다. 아울러 vLLM·SGLang 등 오픈소스 추론 서빙 엔진과 LMCache 등 KV 캐시 최적화 계층이 기존 종속 구조를 부분적으로 완화하는 새로운 변수로 부상하고 있다.

주요국·기업을 3유형 종속 프레임워크로 분석한 결과, NVIDIA는 성능 종속과 구조적 종속의 이중 장벽을, Google은 설계 종속이라는 별도 경로를 구축하고 있으며, 화웨이(Huawei)는 3유형 종속 구조를 자국 내에서 복제·내재화하고 있다.

이 분석 틀을 K-NPU에 적용하면, 한국 NPU 생태계는 '프레임워크 계층 진입에는 성공했으나, 컴파일러·라이브러리 계층의 성능 격차와 운영 생태계 규모 부족이 시장 확산을 제약하는' 구조적 위치에 놓여 있다. PyTorch 네이티브 지원과 vLLM 통합으로 1단계(프레임워크 진입)는 달성하였으나, 2단계(성능 종속 해소)는 진행 중이며, 3단계(운영 생태계 확보)는 초기 단계이다. 특히 2단계의 성능 격차가 좁혀지지 않으면 3단계의 운영 레퍼런스 축적이 곤란하고, 3단계의 레퍼런스가 없으면 2단계의 투자 정당성 확보가 어려운 순환 구조가 존재한다.

이러한 진단에 기반하여 종속 유형별 정책 대응을 제언한다. 첫째, 성능 종속 해소를 위해 칩 설계 중심의 R&D 지원을 컴파일러·런타임·SDK 등 SW 생태계 전반으로 확대하는 HW-SW 균형 발전 패러다임 전환이 필요하다. 둘째, 성능 종속 완화를 위해 PyTorch 호환성 확보와 OpenXLA·MLIR 등 글로벌 오픈소스 표준 참여를 통해 최적화 격차를 협력적으로 축소해야 한다. 셋째, 구조적 종속 우회를 위해 국가 AI 데이터센터 등 공공부문의 실증환경 제공으로 대규모 운영 레퍼런스를 확보하여 순환 구조를 깨야 한다. 넷째, 3유형 종속이 공통적으로 유발하는 전환 비용을 가시화하기 위해 총소유비용(TCO) 기반 평가 체계를 도입해야 한다. 다섯째, 모든 정책의 실행 주체인 AI 컴파일러·시스템 SW 전문 인력 양성 체계를 구축해야 한다.

Global AI spending is projected to reach $2.5 trillion in 2026, with over half flowing into infrastructure. NVIDIA dominates this landscape, capturing roughly 86% of data center GPU revenue. Yet this dominance is not purely a hardware story: on identical H100 chips, software optimization alone can produce over 3x differences in actual throughput. The software ecosystem built over nearly two dec ades since CUDA's 2006 launch constitutes a structural barrier that competitors cannot easily replicate.

This report classifies the AI software stack into four layers—Framework, Compiler, Acceleration Library, and Driver/Runtime—and derives three distinct lock-in mechanisms: performance lock-in, where optimization asymmetries cause de facto convergence toward specific hardware; design lock-in, where framework-compiler-hardware co-design fixes the hardware path at the point of software selection; and structural lock-in, where the closed-source driver/runtime physically blocks hardware substitution. When these types overlap, switching costs increase exponentially. Meanwhile, open-source inference serving engines such as vLLM and SGLang are emerging as new variables that partially mitigate traditional lock-in structures.

Analyzing major players through this framework reveals that NVIDIA maintains a dual barrier of performance and structural lock-in, Google constructs a separate design lock-in pathway through TPU-XLA-JAX, and Huawei replicates all three lock-in types domestically through Ascend-CANN-MindSpore. Applying this framework to K-NPU, we diagnose that Korea's NPU ecosystem has successfully entered the framework layer through PyTorch native support and vLLM integration, but faces a sequential three-stage challenge: Stage 1(framework entry) is achieved, Stage 2(resolving performance lock-in in compiler and library layers) is in progress, and Stage 3(building operational ecosystem scale) remains nascent. A circular dependency exists between Stages 2 and 3—performance gaps hinder reference accumulation, while lack of references undermines investment justification.

Based on this diagnosis, we recommend lock-in-type-specific policy responses: (1) resolving performance lock-in by expanding R&D from chip design to the full software stack; (2) mitigating performance lock-in through participation in global open-source projects such as OpenXLA and MLIR; (3) circumventing structural lock-in by creating public-sector demand to build large-scale operational references that break the circular dependency; (4) introducing a TCO(Total Cost of Ownership) evaluation framework to make switching costs across all three lock-in types visible and quantifiable; and (5) establishing talent pipelines for AI compiler and system software specialists as the execution foundation for all policy measures.

Ⅰ 서론

1. 연구 배경

○ AI 인프라 투자가 전례 없는 규모로 확대되고 있으나, 경쟁의 본질은 하드웨어(HW) 성능을 최대한 활용하게 하는 소프트웨어(SW) 생태계의 종속 구조

- 2026년 전 세계 AI 지출은 2.5조 달러에 이를 전망이며, 그중 절반 이상인 1.37조 달러가 서버·가속기·데이터센터 등 인프라에 집중
  - Gartner(2026.01.)에 따르면 AI 최적화 서버 지출이 전년 대비 49% 증가하며, 이는 전체 AI 시장 지출의 17%를 차지하는 것으로 전망[1]
  - Stanford HAI(2025)에 따르면 2013~2024년 글로벌 AI 누적 투자액 1.6조 달러는 달 착륙 비용과 미 대륙횡단 고속도로 건설비 합산을 초과[2]
  - 이러한 투자의 절대적 규모는 AI 인프라가 반도체·에너지·데이터센터를 아우르는 국가 간 전략 자산으로 전환되고 있음을 시사

- 대규모 투자의 중심에 NVIDIA GPU가 있으며, 데이터센터 GPU 시장에서 경쟁자와의 점유율 격차가 압도적으로 유지되는 구조가 지속
  - NVIDIA는 데이터센터 GPU 매출의 약 86%를 차지하며, AMD(약 10%)·Intel(약 4%)과 현저한 점유율 격차[3]
  - Google TPU, 국내 NPU 등 대안 칩이 등장하고 있으나 범용 시장에서의 생태계 규모와 개발자 기반 측면에서 NVIDIA와의 격차는 현저

- NVIDIA의 지배력은 HW 성능만으로 설명되지 않으며 칩 위에서 작동하는 SW 스택의 최적화 수준이 실질 성능을 결정
  - 스탠포드대학교 연구진은 도메인 특화 아키텍처 시대에는 컴파일러·라이브러리 수준의 SW 최적화가 실질 성능을 좌우한다고 지적[4]

1 Gartner(2026.01.), "Gartner Says Worldwide AI Spending Will Total $2.5 Trillion in 2026"
2 Stanford HAI(2025.04.), "Artificial Intelligence Index Report 2025"
3 Visual Capitalist(2025), "Charted the Battle for AI Data Center Revenue 2021-2025"
4 MLCommons(2024.08.), "New MLPerf Inference v4.1 Benchmark Results Highlight Rapid Hardware and Software Innovations in Generative AI Systems"

- 칩이 동일해도 SW가 다르면 성능이 완전히 달라진다는 점에서, AI 인프라의 경쟁은 반도체를 비롯해 그 위의 SW 계층에서도 진행

○ NVIDIA의 SW 생태계는 약 20년간 축적된 구조적 진입장벽으로 작용

- CUDA는 2006년 출시 이후 약 20년간 라이브러리·개발도구·교육자료·개발자 커뮤니티를 지속적으로 축적하며 독보적 생태계 형성
  - 400만 명 이상의 개발자가 CUDA 생태계에 참여하고 있으며, PyTorch 등 주요 딥러닝 프레임워크와 최적화 수준에서 긴밀하게 통합
  - 이러한 생태계 축적은 경쟁 칩이 유사한 HW 성능을 달성하더라도 전환 비용을 높여 종속(lock-in) 구조를 고착화하는 요인으로 작용

- 유럽집행위원회(EC)와 OECD 등 주요 국제기관도 SW 스택이 AI 인프라 경쟁에서 갖는 구조적 중요성을 공식 분석에서 인정

2. 연구 목적

○ 보고서는 SW가 HW 종속을 형성하는 기술적 경로를 4계층 구조로 추적하고, 종속 메커니즘을 유형화하여 K-NPU의 전략적 포지션을 진단

- 개발자가 프레임워크를 선택하는 순간부터 HW까지의 선택 범위가 점진적으로 좁혀지는 경로 의존성이 종속(lock-in)의 핵심 메커니즘
  - OECD와 NVIDIA의 AI 인프라 SW 스택 분류를 기반으로 '프레임워크 → 컴파일러 → 가속 라이브러리 → HW'로 분류하여 내용을 분석
  - 각 계층에서 종속이 형성되는 기술적 원리를 분석하고, 그것이 HW 대체 가능성과 시장 경쟁 구조에 미치는 정책적 함의를 순차적으로 검토

- NVIDIA, Google, 화웨이, AMD, Intel 등 주요 플레이어의 전략을 검토하고, AI 인프라를 위한 SW 생태계 구축 방향 제언
  - 도출된 분석 틀을 K-NPU에 적용하여 한국 NPU 생태계의 현재 포지션을 종속 유형별로 진단하고, 각 유형에 대응하는 정책 방향을 제언

Ⅱ AI 인프라 SW-HW 5계층 구조

1. AI 인프라의 개념과 분석 범위

○ AI 기술 스택의 주요 기관 정의와 분류

- AI 시스템의 인프라 계층은 HW(GPU, TPU, NPU 등)만으로 구성되지 않으며 그 위에서 작동하는 다층적 SW 구성요소를 포괄하는 구조
  - 동일한 GPU라도 SW 스택에 따라 처리량과 지연·안정성이 크게 달라지며 HW 잠재력을 실제 서비스 성능으로 구현
    * 유럽집행위원회(EC)는 NVIDIA/Run:AI 합병 심사에서 "잘 발달된 SW 스택"의 보유를 데이터센터 GPU 시장의 높은 진입장벽이라 인정[5]

- AI 기술 스택은 분석 목적에 따라서 ① 서비스 관점(AI 밸류체인 전체를 조망) ② 인프라 내부 관점(SW 스택의 기술적 종속 구조)으로 분류될 수 있음
  - **(서비스 관점)** 정책과 규제 논의에서는 주로 AI를 인프라에서 애플리케이션까지 전체 밸류체인을 조망하는 분류가 주로 활용
    * ITI(2025.09)는 4개 기능 계층에 더해 보안·신뢰·규범의 거버넌스가 전 계층을 관통하는 Cross-Layer라 제시
    * Narechania & Sitaraman(2024)은 HW와 클라우드·인프라·모델·앱 4계층으로 시장구조, 지배력, 규율 이슈를 전개
  - 위 분류는 전체적인 조망에는 유용하나 HW 선택에 이르는 SW 내부의 종속 경로를 추적하기에는 해상도가 부족
  - **(인프라 내부 관점)** AI 인프라 내부에서 SW가 HW 종속을 형성하는 경로를 분석하려면 SW 스택 자체의 계층 구조를 더 세밀하게 구분 필요
  - 최종적으로 HW 선택으로 이어지는 종속의 방향성을 추적하여 인프라 선택 시 SW의 역할에 대해 분석
    * 학술영역에서 딥러닝 SW의 워크플로우를 프레임워크 → 컴파일러 → 런타임/시스템 → HW 라이브러리의 4단계로 구분[6]
    * NVIDIA의 CUDA 플랫폼에서도 상위 프레임워크(PyTorch, TensorFlow) → 컴파일러(nvcc, PTX) → GPU 가속 라이브러리(cuDNN, cuBLAS) → 드라이버/런타임 계층 구조로 설계되어 산업계에서도 유사한 분류가 통용

5 European Commission(2024), "Case M.11766 - NVIDIA/Run:AI"

6 Li et al.(2021), "The Deep Learning Compiler: A Comprehensive Survey", IEEE TPDS

분석 범위와 분류 체계

- 우리의 보고서는 AI 인프라 내부에서 SW가 HW 종속을 형성하는 경로를 추적하는 데 목적을 두고, 학술·산업계의 분류 방식을 참고하여 SW 4계층과 HW 계층을 분류한 분석 틀을 채택
  - **(프레임워크)** 개발자가 AI 모델을 설계하는 최상위 도구로 생태계 종속의 출발점으로써 사용 가능한 컴파일러와 라이브러리, HW의 범위를 결정
  - **(컴파일러)** 프레임워크에서 표현된 연산 그래프를 분석하여 연산 융합·메모리 최적화 등 전역 최적화를 수행하고, 최종적으로 HW 타깃에 맞는 실행 코드를 생성하는 계층
  - **(가속 라이브러리)** 특정 HW 아키텍처에 맞게 사전 최적화된 고성능 커널 함수의 집합으로, 프레임워크나 컴파일러가 호출하여 HW의 이론적 성능을 실질 처리량으로 전환하는 계층
  - **(드라이버/런타임)** 운영체제와 HW를 연결하는 최하단 계층으로 메모리 관리, 자원 할당 및 직접적인 기기 제어를 수행
  - **(HW)** 실제 연산이 물리적으로 수행되는 반도체 칩으로 연산 유닛

- 보고서의 4계층 분류는 HW 종속 경로를 분석하기 위한 논리적 구분이며, 실제 구현에서는 계층 간 호출이 반드시 순차적으로 이루어지지 않음
  - 예컨대 PyTorch는 torch.compile을 통해 AI 컴파일러(TorchInductor) 경로를 사용하기도 하지만, cuDNN·cuBLAS 등 가속 라이브러리는 컴파일러를 거치지 않고 직접 호출하는 것 또한 주된 실행 경로

- 다중 경로의 존재는 계층별 종속 강도를 비대칭으로 구성, 우리의 연구는 비대칭성을 분석하기 위해 메커니즘을 기준으로 세 유형의 분석 틀 제시
  - **(성능 종속)** 기술적 대안은 존재하나, 컴파일러의 연산 융합 범위, 가속 라이브러리의 커널 수, 최적화 깊이, 프레임워크의 백엔드별 성능 격차 등 최적화 비대칭으로 인해 사실상 특정 HW로 수렴하는 구조
    * 상위 3개 계층(프레임워크·컴파일러·가속 라이브러리)에서 공통적으로 관찰되는 가장 광범위한 종속 유형
  - **(설계 종속)** 프레임워크-컴파일러-HW가 설계 단계에서 공동 최적화되어, 특정 SW를 선택하는 순간 HW 경로가 확정되는 구조
    * 계층 구분을 횡단하는 교차 경로적 종속으로 성능 종속보다 전환 비용이 높음. JAX → XLA → TPU 수직통합이 대표 사례
  - **(구조적 종속)** 드라이버/런타임 계층의 폐쇄적 구조가 HW 대체를 물리적으로 차단하는 구조
    * 상위 계층이 모두 개방형이더라도 최종 연산은 칩 제조사의 독점 드라이버를 통과해야 실행 가능하며, MIG·NVML 등 데이터센터 운영 기능의 독점이 장벽을 강화

- 이 세 유형은 상호 배타적이 아니며 동일 생태계 내에서 중첩 가능하며 중첩 시 전환 비용은 기하급수적으로 증가
- 2~6절에서 각 계층의 종속 메커니즘을 실증적으로 분석한 뒤, 7절에서 유형 간 상호작용과 중첩 효과를 종합

2. 프레임워크-인프라 종속의 출발점

○ 개발자가 AI 모델을 설계하는 최상위 도구

- **(프레임워크 정의)** 개발자가 AI 모델을 설계하고 학습시킬 때 사용하는 최상위 SW로, 복잡한 HW를 추상화하여 생산성을 높여줌
  - 프레임워크는 텐서(다차원 배열) 연산, 자동 미분, 신경망 구성요소 등 핵심 기능을 표준화된 API로 제공
    * 대표적인 프레임워크: PyTorch(Meta), TensorFlow(Google), JAX(Google), PaddlePaddle(Baidu), MindSpore (Huawei) 등
  - **(내부 구현)** 프레임워크가 없으면 모든 AI 프로젝트에서 자동 미분, 병렬 연산 같은 기본 기능을 매번 새로 구현해야 하므로 개발이 사실상 불가능함
  - 프레임워크 덕분에 개발자는 GPU, TPU, NPU의 복잡한 내부 구조를 몰라도 동일한 코드로 각 칩에서 AI 모델을 학습하고 추론할 수 있음

- **(프레임워크 설계 방식)** '명령형(Imperative)'과 '선언형(Declarative)' 두 가지로 나뉘며, 각각 연구와 서비스 배포 단계에서 서로 다른 장점을 제공
  - **(명령형)** 코드를 한 줄씩 즉시 실행하므로 오류를 바로 확인할 수 있어, 새로운 모델 구조를 빠르게 실험해야 하는 연구 환경에 적합
  - **(선언형)** 전체 연산 구조를 먼저 정의한 뒤 한번에 실행하므로, 전역 최적화가 용이하여 실제 서비스에 배포할 때 성능상 이점 존재

○ 대표 프레임워크

- **(PyTorch)** AI 연구 분야에서 가장 많이 사용되는 프레임워크로 '동적 계산 그래프' 방식 덕분에 일반 Python 코드처럼 자연스럽게 작성 가능

- **(동적 계산 그래프)** 코드가 실행될 때마다 연산 구조가 새로 만들어지므로, Python의 if문이나 for 같은 제어 흐름을 AI 모델 내부에서 사용할 수 있음
- **(정적 그래프)** 정적 그래프는 실행 속도가 빠르지만, 동적 그래프는 유연성이 높아 새로운 아이디어를 빠르게 검증하는 데 유리함[7]

- **(JAX)** Google이 개발한 프레임워크로, 대규모 분산 학습과 고성능 수치 연산에 특화, XLA 컴파일러의 통합으로 TPU 환경에서 최고 수준의 성능 발휘
  - 함수형 프로그래밍 패러다임을 채택하여 동일 입력에 동일 출력을 보장하는 순수 함수 기반으로 설계, 이를 통해 XLA 컴파일러가 전체 연산 그래프를 분석하여 연산 융합·메모리 레이아웃 최적화 등 전역 최적화를 자동 수행
  - jax.jit으로 함수를 XLA 그래프로 컴파일하고, jax.pmap으로 TPU Pod 수천 개 칩에 자동 분산하는 등 컴파일러 중심 설계 철학이 차별점

○ (종속 메커니즘) 프레임워크 계층의 종속은 '선택 불가'가 아닌 '성능 격차에 의한 수렴'이라는 점에서 하위 계층과 질적으로 상이

- **(PyTorch-성능 종속)** PyTorch는 CUDA·ROCm·XLA 등 복수 백엔드를 공식 지원하여 API 수준에서는 HW 중립적이나 cuDNN FlashAttention 등 핵심 가속 기능은 CUDA 경로에서만 완전 동작

- **(JAX-설계 종속)** 설계 단계에서 XLA 컴파일러를 전제하고 구축되어, 프레임워크 선택이 곧 컴파일러(XLA) → HW(TPU) 선택으로 직결되는 설계 종속

7 Stanford University(2025.02.), "CS230 Deep Learning"

[표 Ⅱ-1] 주요 프레임워크 세부사항 정리

항목	PyTorch	TensorFlow	JAX	PaddlePaddle	MindSpore	OneFlow
개발사	Meta	Google	Google	Baidu	Huawei	OneFlow Inc. (칭화대/ 바이트댄스)
출시 연도	2016	2015	2018	2016	2020	2020
라이선스	BSD 3-Clause	Apache 2.0	Apache 2.0	Apache 2.0	Apache 2.0	Apache 2.0
공식 웹사이트	pytorch.org	tensorflow.org	jax. readthedocs.io	paddlepaddle. org.cn	mindspore.cn	oneflow.org
주요 패러다임	명령형	선언형/함수형	함수형	명령형/선언형	명령형	명령형/선언형
코드 스타일	Python native	하이브리드	Python native	Python native	Python native	PyTorch-like
디버깅 난이도	매우 쉬움	중간~어려움	어려움	중간	쉬움	쉬움
그래프 타입	동적 그래프	정적 그래프	동적 함수형	동적 그래프	동적/정적 혼합	동적/정적 혼합
실행 방식	Eager Execution	Eager + 그래프모드	함수형 변환	Eager Execution	하이브리드	Eager Execution + Graph
최적화 수준	중간~높음	높음	매우 높음(TPU)	중간	높음	매우 높음
미분 메커니즘	autograd	GradientTape	함수형 미분	autograd	autograd	autograd
고차 미분	지원	지원	부분 지원	지원	지원	지원
종속 메커니즘 유형	성능 종속	성능 종속 (XLA 경로 시 설계 종속 병존)	설계 종속	성능 종속	설계 종속	성능 종속

PyTorch vs JAX

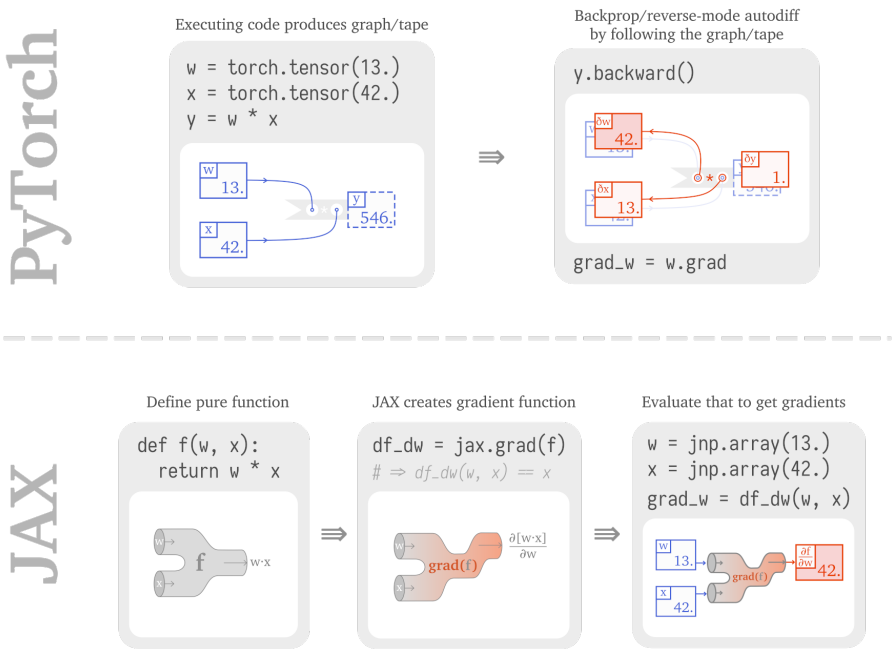
◆ PyTorch와 JAX는 설계 철학과 최적화 접근 방식에서 근본적인 차이

- **(PyTorch)** '명령형 우선' 철학을 채택하여 코드 작성 즉시 연산이 실행되는 Eager Execution을 기본, 일반 Python 코드와 동일하게 동작하여 직관적
  - if문, for/while 루프 등 Python 제어 흐름을 사용할 수 있어 동적 모델 구현에 유리하고, 오류 발생 시 해당 줄에서 확인 가능하여 디버깅이 용이
  - 2.0 버전부터 torch.compile을 도입하여 선택적으로 그래프 컴파일 최적화를 지원하며, Hugging Face 사전학습 모델 라이브러리를 보유

- **(JAX)** '컴파일러 중심' 철학을 채택하여 함수형 프로그래밍과 JIT 컴파일을 핵심으로 설계, Google TPU와 통합되어 대규모 분산 학습에서 고성능 발휘

- XLA 컴파일러를 통한 연산 융합·메모리 최적화 등 전역 최적화를 자동 수행하고, 배치 처리와 다중 디바이스 병렬화를 간결하게 표현
- 순수 함수 패러다임을 따라 동일 입력에 동일 출력을 보장하여 재현성이 높고, 새로운 최적화 알고리즘 연구와 대규모 모델 학습 실험에 적합

[그림 Ⅱ-1] PyTorch와 JAX의 연산 방식 차이



3. 컴파일러(Compiler)와 최적화 도구 - 성능 격차를 만드는 핵심

○ 모델 연산을 중간 표현(IR)으로 변환한 뒤, HW 타깃에 맞게 코드 생성

- 컴파일러는 프레임워크에서 작성된 AI 모델을 칩이 직접 실행할 수 있는 저수준(Low-level) 코드로 변환하며 다양한 성능 최적화를 자동으로 수행

  * 대표적인 AI 컴파일러: XLA(Google), TVM(Apache), Triton(OpenAI), MLIR(LLVM), TensorRT(NVIDIA) 등

- 딥러닝 전용 컴파일러가 등장한 배경은 AI 칩 종류가 급증하면서 각 칩에 맞는 최적화 코드를 일일이 수작업으로 개발하기 어려워졌기 때문

- 딥러닝 컴파일러는 '중간 표현(IR)'이라는 형식과 프론트엔드/백엔드 최적화 기법을 핵심 설계 요소로 활용
- 컴파일러는 서로 다른 프레임워크(PyTorch, TensorFlow, JAX 등)와 다양한 AI 칩(GPU, TPU, NPU) 사이를 연결해주는 SW 다리 역할

  * 딥러닝 컴파일러를 도입하면 프레임워크는 공통된 중간 표현(IR)으로만 변환하고, 각 HW는 IR에서 자기 코드로만 변환하면 되어 효율이 증가

• 딥러닝 컴파일러의 주요 기능 중 하나는 모델을 효율적으로 처리하기 위해 고수준에서 저수준에 이르는 '다층 중간 표현(Multi-level IR)'을 활용
- 고수준 IR에서는 칩 종류와 상관없이 적용 가능한 최적화(불필요한 연산 제거, 연산 순서 조정, 상숫값 미리 계산 등)를 수행
- 저수준 IR에서는 타깃 HW의 특성을 반영하여 메모리 크기, 코어 수, 명령어 특성 등에서 최적화를 수행

• 성능 최적화를 위한 핵심 기술로는 연산 융합(Fusion)을 통해 여러 개의 연속된 딥러닝 연산을 하나로 합쳐서 실행
- 메모리 병목 현상을 해결하고 처리 시간이 획기적으로 감소[8]*

  * NVIDIA cuDNN은 배치 정규화 + ReLU + 컨볼루션을 융합하여 메모리 읽기/쓰기 횟수를 줄이고 처리 속도를 향상시키는 대표적 사례

• 컴파일러 계층에서 성능 격차가 발생하는 기술적 원인
- **(연산 융합 범위의 차이)** XLA는 전체 프로그램 수준 융합이 가능한 반면, 범용 컴파일러는 로컬 패턴에 한정
- **(자동 튜닝 탐색 공간의 규모)** TVM의 AutoTVM은 수십억 개 구성을 탐색하나, HW별 탐색 효율이 상이
- **(중간 표현(IR) 표현력의 차이)** MLIR 기반 컴파일러는 다층 IR로 HW 특성을 정밀 반영 가능

  * 이러한 기술적 차이가 동일 칩에서도 1.2~3.8배의 성능 격차로 귀결, 특정 HW에 대한 컴파일러 최적화 투자가 곧 해당 HW의 실질 경쟁력을 결정짓는 구조 형성[9]

- 이는 1절에서 정의한 '성능 종속'의 컴파일러 계층 발현으로, 대안 컴파일러가 존재하나 최적화 격차가 사실상의 HW 수렴을 유발하는 구조

---

8 덧셈, 곱셈, 활성화 함수를 각각 따로 실행하면 중간 결과를 매번 메모리에 저장했다가 다시 읽어야 해서 시간과 전력이 낭비되기에 연산을 하나로 합치면 중간 결과가 칩 내부의 빠른 레지스터에만 저장되고, 느린 외부 메모리 접근 횟수를 줄일 수 있어서 처리 속도를 크게 향상시킬 수 있음

9 Tianqi Chen et al.(2018.02.), "TVM: An Automated End-to-End Optimizing Compiler for Deep Learning"

[표 Ⅱ-2] 주요 AI 컴파일러 세부사항 정리

항목	XLA	TorchInductor	TVM(Apache)	TensorRT	ONNX Runtime(ORT)
운영 주체 (생태계)	Google(TF/JAX)	PyTorch (Meta/커뮤니티)	Apache 커뮤니티	NVIDIA	MS/커뮤니티
주요 역할	학습·추론 그래프 컴파일(JIT/AOT)	PyTorch 2.x 기본 컴파일러	범용 딥러닝 컴파일러(학습·추론)	추론 컴파일러/ 엔진(엔진 빌드)	ONNX 실행+그래프 최적화(EP 기반)
핵심 기술 (대표)	HLO 기반 최적화, fusion, 스케줄링	FX graph → 최적화, 커널 생성 (주로 Triton)	Relay/IR, AutoTVM/Ansor (자동튜닝), 코드생성	레이어 fusion, precision(INT8/ FP16), 커널 선택	그래프 최적화, Execution Provider (TensorRT/CUDA/ ROCm 등)
주요 타깃	TPU/CPU/GPU	NVIDIA/AMD GPU, CPU	CPU/GPU/엣지/ 가속기	NVIDIA GPU	CPU/GPU/ 가속기(EP)
장점	TF/JAX에 자연스러운 통합, TPU 강점	PyTorch 사용성 유지하며 성능 향상	HW 다양성·이식성, 연구/산업 적용 폭넓음	추론 성능 최상위권, 운영 레퍼런스 많음	ONNX 중심 상호 운용성, 다양한 EP
한계/주의	PyTorch 1st-class는 아님(직접 핵심 경로 X)	백엔드/타깃별 성숙도 차이, 디버깅 난이도	파이프라인 구성 난이도, 제품 통합 비용	NVIDIA 종속, 학습 범용 컴파일러로는 부적합	"완전한 단일 컴파일러"라기보다 컴파일+런타임 혼합

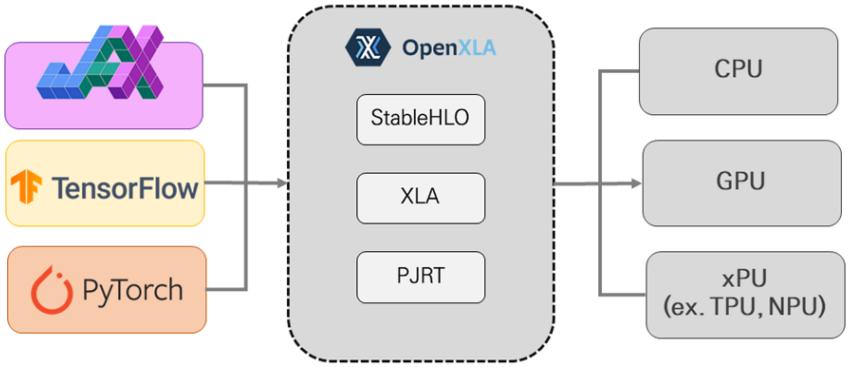
Google의 수직통합 전략 **TPU-XLA-JAX의 긴밀한 연결**

◆ XLA는 Google이 자사 TPU의 성능을 극대화하기 위해 설계한 컴파일러로, HW와 컴파일러를 동시에 개발하는 '공동 설계' 전략의 산물

- **(TPU 최적화 설계)** TPU는 범용 GPU와 달리 행렬 연산에 특화된 구조로, XLA는 TPU의 시스톨릭 어레이 아키텍처에 맞춰 연산 그래프를 최적 배치
  - XLA는 TPU Pod 수천 개 칩을 단일 거대 장치처럼 프로그래밍할 수 있게 하며, 컴파일러가 자동으로 모델과 텐서를 분할하고 통신 연산을 삽입
  - 커널 퓨전·메모리 레이아웃 최적화·지연 숨김 스케줄링 등 TPU 특성에 맞춘 최적화를 통해 GPU 대비 동일 연산에서 높은 전력 효율 달성

- **(JAX의 역할)** JAX는 XLA를 백엔드로 설계한 프레임워크로, jax.jit으로 함수를 XLA 그래프로 컴파일하고 jax.pmap으로 TPU 전체에 자동 분산
  - 순수 함수 패러다임과 컴파일러 중심 설계로 인해 XLA가 전역 최적화를 수행하기에 이상적이며, TPU에서 훨씬 높은 성능을 발휘함
  - vLLM TPU 백엔드는 PyTorch로 작성된 모델도 JAX 경로로 변환하여 XLA 컴파일을 수행하며, 동일 모델에서 20% 이상의 추가 성능 향상

[그림 Ⅱ-2] OpenXLA SW 에코시스템



출처: OpenXLA 공식 홈페이지 내용 SPRi 재구성

Triton 커널 프로그래밍의 다변화와 가속 라이브러리 종속 완화 가능성

◆ OpenAI가 개발한 Triton은 GPU 커널을 Python으로 작성할 수 있게 하는 컴파일러 언어로, cuDNN 등 벤더 제공 라이브러리에 대한 종속을 부분적으로 완화하는 새로운 흐름을 형성

- **(등장 배경)** 기존 CUDA 커널 작성은 스레드 블록 구성, 공유 메모리 관리 등 GPU 아키텍처에 대한 깊은 전문성을 요구하며, 새로운 연산 패턴이 등장할 때마다 벤더의 라이브러리 업데이트를 기다려야 하는 구조적 병목
  - Triton은 블록 단위 프로그래밍 모델을 채택하여, 개발자가 타일 크기와 연산 논리만 정의하면 컴파일러가 메모리 접근 패턴과 명령어 스케줄링을 자동 최적화
  - Tillet et al.(2019)에 따르면, 전문가 수준 CUDA 커널 대비 90% 이상의 성능을 달성하면서 코드량을 1/10 수준으로 절감

- **(PyTorch 통합과 생태계 효과)** PyTorch 2.0의 torch.compile 기본 백엔드인 TorchInductor가 Triton을 커널 생성 엔진으로 활용하면서 사실상 주류 AI 개발 파이프라인에 편입
  - FlashAttention(Dao et al., 2022)은 Triton으로 구현된 대표 사례로, cuDNN 기존 어텐션 대비 2~4배 빠른 속도를 달성하여 LLM 학습·추론의 사실상 표준
  - 이는 벤더 라이브러리 업데이트를 기다리지 않고 연구자가 직접 고성능 커널을 작성할 수 있음을 보여주며, 본 보고서가 분석한 '가속 라이브러리 종속' 경로를 부분적으로 약화시키는 변수

- **(한계)** 현재 Triton 백엔드는 NVIDIA GPU에 가장 성숙하며, 최종 코드 생성 타깃이 특정 HW의 명령어 체계(ISA)인 점에서 드라이버/런타임 계층의 종속 자체 해소 불가
  - 다만 Triton 백엔드가 다양한 HW로 확장될 경우 동일 코드로 여러 칩을 타깃할 수 있어 중장기적으로 HW 전환 비용을 낮추는 방향으로 작용 가능

4. 가속 라이브러리 - 칩의 잠재력을 현실화하는 SW

○ 컴파일러가 변환·최적화한 코드를 실제 칩 위에서 구동하기 위해 필요한 SW 환경

○ (HW 추상화 라이브러리) 칩의 고유 기능을 표준 인터페이스로 제공

- cuDNN·cuBLAS 등 최적화 라이브러리는 컨볼루션·행렬곱·어텐션 등 핵심 연산을 각 칩의 HW 특성에 맞게 극도로 최적화한 커널 함수의 집합

- 상위 SW는 칩의 세부 구조를 몰라도, 이 라이브러리가 제공하는 표준 함수 호출만으로 해당 칩의 잠재 성능을 최대한 끌어낼 수 있는 추상화 구조
  - NVIDIA cuDNN은 GPU 기반 딥러닝의 사실상 표준으로, PyTorch·TensorFlow 등 주요 프레임워크가 내부에서 자동 호출하는 필수 의존 관계
  - cuBLAS는 행렬 곱셈(GEMM) 등 기초 선형대수 연산을 GPU에서 처리하는 라이브러리로 순전파와 역전파 모두 대규모 행렬 연산의 연속이므로 전체 AI 처리 속도의 하한선을 결정

  * 기초 라이브러리의 최적화 수준이 높을수록 그 위에서 동작하는 모든 프레임워크와 모델이 자동으로 성능 향상 혜택을 받는 누적적 구조

- AMD의 MIOpen·rocBLAS, Intel의 oneDNN·oneMKL은 커널 수와 최적화 깊이에서 NVIDIA의 cuDNN, cuBLAS 대비 성숙도 격차가 존재

(분산 통신 라이브러리) 대규모 학습의 효율을 결정

- NCCL(NVIDIA Collective Communications Library)은 다수 GPU 간 학습 중 계산된 기울기 값을 초고속으로 동기화하는 통신 라이브러리
  - 수천 개 GPU를 활용하는 대규모 모델 학습에서 통신 효율이 전체 학습 시간과 비용을 직접적으로 좌우

- NCCL은 NVLink·NVSwitch와 긴밀히 통합 설계되어, HW 인터커넥트와 SW가 공동 최적화된 결과물이자 종속 체인을 더욱 강화하는 요소

- AMD의 RCCL이 NCCL 호환 API를 제공하여 마이그레이션 용이성을 추구하나, HW-SW 공동 설계의 깊이에서 차이
  - 텐서 코어를 최대 활용하도록 수년간 정밀 튜닝된 결과이며, 입력 크기·형태에 따라 최적 알고리즘을 자동 선택하는 휴리스틱을 기본 탑재

(추론 서빙 SW) LLM 서비스의 경제성을 좌우하는 추론 서빙 계층에서 오픈소스 SW 혁신이 가속화되며, AI 인프라 경쟁의 새로운 전선을 형성

- **(추론 서빙의 전략적 중요성)** AI 워크로드에서 추론이 차지하는 비중이 학습 대비 빠르게 증가하면서, 추론 서빙 SW의 효율성이 곧 인프라 총소유비용(TCO)을 결정하는 핵심 변수로 부상
  - Gartner는 AI 추론 시장이 갈수록 확대되고 있으며 GPU 외에도 ASIC·TPU 등 다양한 가속기 투자가 확대 중임을 분석[10]
    * 학습은 일회성 비용 구조인 반면, 추론은 서비스 중 지속적으로 누적되는 비용 구조이므로 추론 효율 1%p 향상이 연간 수억 원 규모의 운영비 절감으로 직결[11]

- **(vLLM·SGLang 양강 구도)** '25년 기준 LLM 추론 서빙 엔진은 vLLM과 SGLang이 사실상의 오픈소스 표준으로 자리매김
  - vLLM은 PagedAttention 기반 메모리 관리로 GPU 메모리 활용 효율을 극대화하며[12], 업계 최대 규모의 커뮤니티를 보유[13]

10 Gartner(2025.10.10.), "Gartner Says AI-Optimized IaaS Is Poised to Become the Next Growth Engine for AI Infrastructure"
11 NVIDIA(2025.06.18.), "LLM Inference Benchmarking: How Much Does Your LLM Inference Cost?"
12 Kwon, Woosuk, et al.(2023), "Efficient Memory Management for Large Language Model Serving with PagedAttention"
13 PyTorch(2024.09.), "vLLM Joins PyTorch Ecosystem: Easy, Fast, and Cheap LLM Serving for Everyone"

- SGLang은 RadixAttention 기반 메모리 관리와 Zero-Overhead 배치 스케줄러로 vLLM 대비 약 29% 높은 처리량을 H100 벤치마크에서 달성[14]

  * SGLang은 전 세계 40만 개 이상의 GPU에서 운용되며, '25.10월 TPU에서도 JAX 백엔드를 통한 네이티브 실행을 지원하기 시작[15]

- 두 엔진 모두 NVIDIA GPU 중심으로 최적화되어 있으나, vLLM은 AMD GPU·TPU·Ascend 등 다중 HW 백엔드를 플러그인 구조로 지원하며, SGLang 역시 AMD GPU·Ascend NPU 등으로 HW 지원을 확장 중

- **(KV 캐시 계층의 등장)** LMCache 등 KV 캐시 전문 SW가 추론 서빙의 새로운 최적화 계층으로 부상하며, 추론 SW 스택의 세분화가 진행
  - LMCache는 추론 엔진(vLLM, SGLang)과 스토리지 백엔드 사이에 위치하여 KV 캐시를 GPU 메모리 외부(CPU, 디스크, Redis, RDMA)로 오프로딩하고 엔진 간 공유를 지원
  - vLLM과 결합 시 다중 질의응답·문서분석 등 워크로드에서 최대 15배 처리량 향상을 달성하며, '25년 PyTorch 공식 생태계에 편입[16]

    * Google Cloud[17], CoreWeave[18], NVIDIA Dynamo[19] 등 주요 인프라 사업자가 LMCache를 채택하여 프로덕션 환경에서 활용 중

- 가속 라이브러리 계층의 종속은 '커널 수와 최적화 깊이의 비대칭'에서 비롯
  - NVIDIA cuDNN은 입력 크기·형태·정밀도에 따라 최적 알고리즘을 자동 선택하는 휴리스틱을 탑재하고 수천 개 수준의 사전 최적화 커널을 보유
  - AMD MIOpen·Intel oneDNN은 지원 연산 종류, 최적화 경로에서 격차
  - 프레임워크가 이들 라이브러리를 컴파일러 없이 직접 호출하는 경우도 많아 라이브러리의 성숙도 차이가 곧 프레임워크의 HW별 성능 차이로 직결되는 구조적 특성
  - 가속 라이브러리 계층의 종속 역시 1절의 '성능 종속' 유형에 해당하며, 커널 수와 최적화 깊이의 비대칭이 중요

14 AIMultiple(2026.01.), "LLM Inference Engines: vLLM vs LMDeploy vs SGLang"
15 SGLang(2026), "SGLang Documentation"
16 Liu, Yuhan et al.(2025), "LMCache: An Efficient KV Cache Layer for Enterprise-Scale LLM Inference"
17 Google Cloud(2025), "Boosting LLM Performance With Tiered KV Cache on Google Kubernetes Engine"
18 CoreWeave(2025.10.16.), "CoreWeave Unveils AI Object Storage, Redefining How AI Workloads Access and Scale Data"
19 NVIDIA(2026.02.19.), "LMCache Integration in Dynamo"

[표 II-3] 주요 AI 디바이스 라이브러리 세부사항 정리

분류	구분	라이브러리	주요 역할	대응/비고 (NVIDIA 기준)	특징 및 설명	종속 메커니즘
연산 가속	NVIDIA	cuDNN	딥러닝 연산 가속	(기준)	Conv, Pooling 등 핵심 연산 최적화, 프레임워크 필수 요소	성능 종속 형성—수천 개 최적화 커널이 대안 대비 비대칭적 성능 우위를 구축하여 HW 수렴 유도
		cuBLAS	선형대수 연산 가속	(기준)	행렬 곱셈(GEMM) 등 모든 AI/HPC 연산의 기초 제공	성능 종속 형성—GEMM 최적화 깊이가 전체 스택 성능의 기저를 결정
	AMD	MIOpen	딥러닝 연산 가속	cuDNN 대응	ROCm 플랫폼 기반, 오픈소스이며 튜닝 기능 제공	성능 종속 완화 시도—오픈소스 접근으로 커널 격차 축소를 지향하나, 커널 수·최적화 깊이에서 cuDNN 대비 격차 잔존
		rocBLAS	선형대수 연산 가속	cuBLAS 대응	HIP 언어로 작성되어 CUDA API와 유사한 구조	성능 종속 완화 시도—CUDA API 호환 구조로 전환 비용 저감을 지향
	Intel	oneDNN	딥러닝 연산 가속	cuDNN 대응	CPU(AVX-512) 및 인텔 GPU(Arc) 통합 최적화	성능 종속 완화 시도—CPU·GPU 통합 최적화로 NVIDIA 외 경로 확보를 지향
		oneMKL	수학/과학 연산 가속	cuBLAS 등 포괄	AI뿐만 아니라 과학, 엔지니어링 전반의 고성능 커널 제공	성능 종속 완화 시도—범용 수학 커널로 HW 중립적 성능 기반 구축
	NPU	TT-NN	딥러닝 연산	cuDNN 대응	Tenstorrent 칩 전용, HW 접근성이 높은 개방형 구조	설계 종속(자체 경로)—칩 전용 라이브러리로 SW 선택이 곧 HW를 확정하나, 개방형 설계로 종속 강도를 의도적으로 완화
		PopLibs	연산 및 통신	cuDNN + NCCL 대응	Graphcore IPU의 그래프 처리 구조에 최적화된 라이브러리군	설계 종속(자체 경로)—IPU 아키텍처 전용 최적화로 라이브러리 선택이 HW 경로를 확정하는 수직통합 구조
분산 통신	NVIDIA	NCCL	멀티 GPU 통신	(기준)	분산 학습 시 GPU 간 초고속 데이터 전송 (NVLink 활용)	성능 종속+구조적 종속 중첩—NVLink HW 종속과 결합하여 통신 계층에서 이중 장벽 형성
	AMD	RCCL	멀티 GPU 통신	NCCL 대응	NCCL과 호환되는 API 제공으로 마이그레이션 용이	성능 종속 완화 시도—API 호환으로 전환 비용을 저감하나 NVLink 대비 통신 성능 격차 잔존
추론 서빙	NVIDIA	TensorRT	추론 실행	(기준)	프로덕션 환경용 경량 런타임, 메모리 및 지연시간 최적화	성능 종속 형성—NVIDIA GPU 전용 추론 최적화로 프로덕션 환경의 HW 수렴을 고착화
	오픈소스	vLLM	LLM 추론 서빙	범용	PagedAttention 기반, 다양한 HW 백엔드 지원 확장 중	성능 종속 완화 경로—HW 중립적 설계로 추론 계층의 종속을 해소하는 개방형 대안
	NPU	Furiosa/ RBLN SDK	추론 런타임	TensorRT 대응	국내 NPU 진영, ONNX/vLLM 지원 및 NPU-CPU 최적화	성능 종속 우회 진입—ONNX·vLLM 호환으로 프레임워크 계층 진입장벽을 우회하되, 커널 최적화 깊이 확보가 과제

5. 드라이버 및 런타임 - HW 제어와 생태계 종속성의 완성점

○ 상위 라이브러리가 호출한 연산 명령을 물리적 기계가 직접 실행할 수 있도록 HW를 제어하는 최하단 시스템 계층

- **(디바이스 제어 및 자원 스케줄링)** 칩의 물리적 실행 환경을 직접 구성
  - 운영체제 커널 수준에서 동작하며 상위 스택이 요구하는 메모리 동적 할당, 코어별 스레드 스케줄링, 통신 채널 초기화 등 필수적인 물리적 실행 환경 구축
  - 디바이스 메모리 할당, 코어별 스레드 스케줄링, 통신 채널 초기화 등 AI 연산에 필요한 필수적인 물리적 실행 환경을 운영체제 하단에서 통제
  - 운영체제와 AI 가속기 사이에서 자원 충돌을 방지하고, 칩의 전력 소모와 온도를 실시간으로 제어하여 대규모 인프라의 구동 안정성을 직접 보장

    * NVIDIA의 디바이스 드라이버와 NVML(NVIDIA Management Library)은 GPU 상태를 추적하고 관리[20]

- **(가상화 및 인프라 효율 극대화)** 데이터센터 수준의 HW 활용도 결정
  - NVIDIA의 MIG(Multi-Instance GPU) 같은 런타임 기술은 단일 물리 칩을 여러 개의 논리적 인스턴스로 안전하게 분할하여 제공[21]
  - 수많은 AI 워크로드가 혼재된 클라우드 환경에서 HW 자원을 엄격히 격리하고 배분하는 런타임 성능이 곧 데이터센터의 서비스 경제성으로 직결

- **(호스트-디바이스 간 통신 및 동기화)** 분산 학습 물리 계층의 병목 해소
  - 상위의 NCCL 같은 분산 통신 라이브러리가 제 성능을 내기 위해서는 최하단 드라이버가 NVLink 등의 물리 계층을 직접 제어하며 트래픽을 라우팅해야 할 필요
  - CPU(호스트)와 GPU(디바이스) 간 데이터 이동 병목을 최소화하기 위해 런타임 계층은 PCIe나 CXL 인터페이스 위에서 초고속 데이터 전송을 조율
  - AMD의 ROCm 런타임 역시 CPU와 GPU가 통합된 메모리 공간을 공유할 수 있도록 지원하여 통신 오버헤드를 줄이는 방향으로 생태계를 발전 중[22]

---

20 NVIDIA Management Library(NVML)

21 NVIDIA Multi-Instance GPU User Guide

22 ROCm: Unified memory management

- **(수직통합과 종속성 심화)** HW 선택을 강제하는 근본적인 Lock-in 구조
  - 프레임워크나 컴파일러 등 상위 스택에서 개방형 표준을 채택하더라도, 최종 연산은 반드시 칩 제조사가 독점 배포하는 폐쇄적인 드라이버를 통과
  - 상위 계층에서 범용적인 최적화를 수행하더라도 최종 처리량과 지연 시간은 제조사의 고유 아키텍처와 결합된 드라이버의 스케줄링 능력에 의존
  - CUDA 드라이버처럼 자사 칩 아키텍처의 세밀한 변화에 맞춰 극도로 튜닝된 런타임 환경은 경쟁사가 단기간에 모방할 수 없는 구조적 장벽 형성[23]
    * CUDA 드라이버는 비공개 소스(Closed Source)로 배포되어 경쟁사가 동등한 기능을 리버스 엔지니어링으로 구현하는 것이 기술적·법적으로 차단
  - 사용자는 예측 불가능한 시스템 오류를 방지하고 인프라 안정성을 담보하기 위해, 최하단 SW가 가장 검증된 특정 HW를 선택하는 구조
    * MIG·NVML 등 데이터센터 운영에 필수적인 가상화·모니터링 기능이 NVIDIA 드라이버에만 내장되어 있어, 대안 HW 도입 시 운영 인프라 전체를 재구축 부담
  - 이는 '구조적 종속'의 핵심 발현으로, 성능 종속과 달리 최적화 격차의 축소가 아닌 폐쇄적 소스 구조 자체가 전환을 차단하는 메커니즘

[표 Ⅳ-4] CUDA SW 스택

분류	구분	드라이버/런타임	주요 역할	대응/비고 (NVIDIA 기준)	특징 및 설명	종속 메커니즘
GPU 드라이버	NVIDIA	CUDA Driver	GPU 제어 및 커널 실행	(기준)	비공개 소스, 커널 모드 드라이버로 GPU HW 직접 제어. 모든 상위 SW 스택의 최종 실행 경로	구조적 종속 형성(핵심) — 폐쇄적 소스 구조가 HW 대체를 물리적으로 차단하는 종속의 최하단 고정점
	AMD	ROCm Driver (amdgpu)	GPU 제어 및 커널 실행	CUDA Driver 대응	오픈소스(Linux 커널 통합), HIP 런타임과 연동	구조적 종속 완화 — 오픈소스 드라이버로 투명성을 확보하여 NVIDIA의 폐쇄적 구조와 차별화
	Huawei	Ascend Driver (npu-drv)	Ascend NPU 제어	CUDA Driver 대응	비공개 소스, Ascend 310/910 시리즈 전용 커널 모드 드라이버. CANN 스택의 최하단 실행 경로	구조적 종속 형성(자국 내) — NVIDIA와 동일한 폐쇄적 드라이버 구조를 Ascend 생태계에서 복제하여 자국 내 HW 경로를 고정
	Baidu	Kunlun Driver (xpu-drv)	Kunlun XPU 제어	CUDA Driver 대응	Kunlun 1세대/2세대 칩 전용 드라이버, 비공개 소스	구조적 종속 형성(자국 내) — Kunlun 칩 전용 폐쇄적 드라이버로 자체 HW 경로를 확보하나, 생태계 규모가 화웨이 대비 제한적
런타임	NVIDIA	CUDA Runtime	API 추상화 및 실행 관리	(기준)	메모리 할당·스트림 관리·커널 런칭 등 프로그래밍 인터페이스 제공. 사실상 AI 개발의 표준 런타임	성능 종속+구조적 종속 중첩 — 광범위한 API 생태계(성능 종속)와 폐쇄적 드라이버 연동(구조적 종속)이 동시 작동

23 European Commission(2024), "Case M.11766 - NVIDIA/Run:AI"

분류	구분	드라이버/런타임	주요 역할	대응/비고 (NVIDIA 기준)	특징 및 설명	종속 메커니즘
런타임	AMD	HIP Runtime	API 추상화 및 실행 관리	CUDA Runtime 대응	CUDA API와 구문 수준 호환, hipify 도구로 CUDA 코드 자동 변환 지원	성능 종속 완화 시도 — CUDA 호환 API로 전환 비용을 저감하나, 변환 완전성·성능 동등성에서 격차 잔존
	Intel	SYCL	상위 프로그래밍 모델	CUDA 대응 (상위 계층)	ISO C++ 기반 개방형 표준, 단일 소스로 이기종 HW 대응	성능 종속 완화 시도 — 개방형 표준으로 특정 HW 종속을 회피하나, CUDA 생태계 대비 라이브러리·도구 규모 부족
	Google	libtpu (TPU Runtime)	TPU 제어 및 실행 관리	해당 없음 (독자 경로)	XLA 컴파일러 출력을 TPU에서 실행하는 전용 런타임, 비공개 소스	설계 종속(경로 확정) — JAX → XLA → libtpu → TPU 수직통합의 최하단으로, 설계 종속 경로의 실행 계층
	Huawei	CANN Runtime	Ascend NPU 제어 및 실행	CUDA Runtime 대응	Ascend 칩 전용 런타임, 자체 연산자 라이브러리(AscendCL)·그래프 엔진 내장. MindSpore·PyTorch·TensorFlow 백엔드 지원	설계 종속+구조적 종속 복제 — MindSpore → CANN → Ascend 수직통합으로 NVIDIA의 종속 구조를 자국 내 재현. 설계 종속 경로와 폐쇄적 드라이버 구조가 동시 작동
	Huawei	AscendCL	프로그래밍 인터페이스	CUDA Runtime API 대응	CANN 상위의 C/C++ API 계층, 연산자·메모리·스트림 관리. CUDA API와 유사한 구조이나 호환성은 없음	성능 종속 형성(자국 내) — 독자 API 생태계를 구축하여 Ascend 내 개발자 종속을 유도하나, CUDA 대비 서드파티 라이브러리·도구 규모에서 격차
	Baidu	Kunlun XRE (XPU Runtime Engine)	Kunlun XPU 제어 및 실행	CUDA Runtime 대응	Kunlun 칩 전용 런타임, XTDK(개발 도구킷)와 통합. PaddlePaddle과 긴밀 연동 설계	설계 종속(자국 내 경로) — Paddle Paddle → XTCL 컴파일러 → XRE → Kunlun 수직통합으로, 프레임워크 선택이 HW 경로를 확정하는 구조. 다만 PaddlePaddle의 CUDA 경로 병존으로 종속 강도는 화웨이 대비 약함
	Furiosa	Furiosa SDK Runtime	NPU 제어 및 실행	CUDA Runtime 대응	ONNX 모델 직접 실행, 컴파일러-런타임 통합 구조	성능 종속 우회 진입 — ONNX 호환으로 프레임워크 종속을 우회하되, 자체 런타임 생태계 규모 확보가 과제
	Rebellions	RBLN Runtime	NPU 제어 및 실행	CUDA Runtime 대응	vLLM 백엔드 통합, PyTorch 네이티브 지원	성능 종속 우회 진입 — 기존 SW 생태계(vLLM·PyTorch)와의 호환으로 진입장벽을 우회하는 전략
데이터센터 관리	NVIDIA	NVML/ nvidia-smi	GPU 모니터링·관리	(기준)	온도·전력·메모리 사용량 실시간 모니터링, 데이터센터 운영의 사실상 표준	구조적 종속 강화 — HW 전환 시 운영 도구 전체를 교체해야 하는 관리 계층 종속 형성
	NVIDIA	MIG	GPU 자원 분할	(기준, 독점)	단일 GPU를 최대 7개 독립 인스턴스로 분할, A100/H100 이상 지원. 경쟁사 동등 기능 부재	구조적 종속 강화(핵심) — 멀티테넌트 운영의 필수 기능을 독점하여, 성능 격차와 무관하게 운영 수준에서 전환을 차단
	AMD	ROCm SMI	GPU 모니터링·관리	NVML 대응	오픈소스 모니터링 도구, 기본 기능 제공	구조적 종속 완화(부분적) — 기본 모니터링은 대응하나 MIG 동등 기능 부재로 데이터센터 운영 전환에 한계
	Huawei	MindX DL/ Ascend Device Plugin	Ascend NPU 관리·스케줄링	NVML+MIG 대응	Kubernetes 기반 Ascend 자원 관리·모니터링·가상 분할(VNPU) 지원. 중국 내 AI 데이터센터 운영 표준 지향	구조적 종속 형성(자국 내) — VNPU (가상 NPU)로 MIG 동등 기능을 구현하여 관리 계층까지 포함한 완결적 종속 구조를 복제. 중국 내 Ascend 전환 장벽 형성
	Baidu	Kunlun Management Tool	Kunlun XPU 모니터링	NVML 대응	Kunlun 칩 상태·온도·전력 기본 모니터링 기능 제공	구조적 종속 형성(초기) — 기본 관리 기능은 확보하였으나, MIG/VNPU 동등의 자원 분할 기능은 미구현으로 데이터센터 운영 종속 강도가 화웨이 대비 약함

6. HW-SW가 성능을 실현하는 대상

○ 실제 연산이 물리적으로 일어나는 반도체 칩

- HW 계층은 AI 연산을 물리적으로 수행하는 반도체 칩으로, GPU, TPU, NPU 등 각기 다른 설계 철학과 아키텍처를 가진 다양한 제품들이 경쟁 중임
  * 주요 AI HW: NVIDIA GPU(H100, B200 등), Google TPU, AMD Instinct, Intel Gaudi, 국내 NPU(리벨리온 ATOM, 퓨리오사 RNGD) 등

- 각 칩은 고유한 명령어 체계(ISA), 메모리 계층 구조, 연산 유닛 배치를 가지며, 해당 칩에 맞게 최적화된 SW가 있어야 성능을 제대로 발휘함
  - 칩의 이론적 연산 처리 능력(FLOPS), 메모리 대역폭, 전력 효율은 AI 작업의 처리 속도와 운영 비용을 결정하는 물리적 상한선을 형성함
  - 그러나 이론적 성능 수치와 실제 워크로드에서의 성능은 SW 최적화 수준에 따라 상당한 차이를 보이는 경우가 많음
  * UC 버클리의 데이비드 패터슨(David Patterson) 교수는 "현재 도메인 특화 아키텍처의 성능 향상은 HW보다 SW 스택 최적화가 결정하며 동일 칩에서도 컴파일러 수준에 따라 10배에서 100배 이상 성능 차이가 발생할 수 있다"고 발언[24]

- AI 칩 설계에서 가장 큰 도전 과제는 '메모리 병목(Memory Wall)' 문제로, 연산 장치가 데이터를 기다리느라 유휴 상태로 낭비되는 시간이 많다는 점임
  - 최신 AI 모델, 특히 GPT 같은 언어모델은 어텐션 연산이 핵심인데, 이 연산은 대량의 데이터를 메모리에서 읽고 써야 해서 메모리 병목이 심각

- 시스톨릭 어레이(Systolic Array)는 Google TPU가 채택한 대표적인 AI 칩 아키텍처로, 행렬 연산에 특화되어 데이터가 규칙적으로 흐르며 재사용됨
  - 데이터가 여러 연산 장치(PE: Processing Element)를 순차적으로 지나가며 계속 재사용되므로, 외부 메모리 접근 횟수가 크게 줄어듦
  - 컨볼루션이나 행렬곱 같은 AI 핵심 연산에서 높은 효율을 보여주어, 최근 출시되는 많은 NPU 설계에서 유사한 데이터 흐름 방식을 채택하고 있음

24 ACM SIGARCH(2018.04.), "John Hennessy and David Patterson Share ACM Turing Award"

- HBM(High Bandwidth Memory, 고대역폭 메모리)은 메모리 칩을 수직으로 쌓아 기존 DDR 대비 수배 빠른 데이터 전송 속도를 제공하는 핵심 부품임
  - HBM은 GPU나 NPU 패키지 안에 함께 들어가 신호 이동 거리가 짧아지고 데이터 통로가 넓어져서 빠른 속도와 낮은 전력 소비를 동시에 달성함

- HW 성능 수치만으로 AI 칩의 실제 경쟁력을 평가할 수 없는 이유는 SW 스택의 성숙도가 칩 활용률을 근본적으로 결정하기 때문

7. 계층 간 상호작용: HW-SW 공동 설계가 경쟁력을 결정

○ (종속 강도의 비대칭성) SW 4계층의 종속 효과는 균일하지 않으며, 계층별로 종속의 유형·강도·완화 가능성이 구조적으로 상이

- 상위 프레임워크부터 최하단 드라이버까지 4계층이 유기적으로 결합해야 물리적 HW의 잠재력을 온전히 활용 가능

- **(프레임워크)** 형식적 개방성과 사실상의 성능 종속이 공존하는 '비대칭 종속'
  - PyTorch는 CUDA·ROCm·XLA 등 복수 백엔드를 공식 지원하여 프레임워크 선택 자체가 특정 HW를 강제하지는 않음
  - 그러나 CUDA 경로에서만 cuDNN FlashAttention 등 핵심 가속 기능이 완전히 동작, 비CUDA 경로에서는 동일 모델의 처리량 저하
    * PyTorch 공식 문서에서도 CUDA 백엔드가 "가장 성숙한 경로(most mature path)"로 명시되어 있어, 기능적 개방성이 성능적 대등함을 의미하지 않음
  - 따라서 프레임워크 계층의 종속은 HW 선택을 강제하는 구조적 종속이 아닌 최적 성능을 위해 특정 HW로 수렴하게 만드는 성능 종속으로 분류 가능

- **(프레임워크-HW 공동 설계 경로)** JAX → XLA → TPU는 프레임워크 선택이 곧 HW 선택으로 직결되는 강한 경로 종속의 대표 사례
  - JAX는 설계 단계에서부터 XLA 컴파일러를 백엔드로 전제하고 구축되었으며, jax.jit이 함수를 XLA HLO 그래프로 변환하는 것이 핵심 실행 경로

- XLA는 TPU의 시스톨릭 어레이 아키텍처에 맞춰 연산 배치·메모리 레이아웃·통신 스케줄링을 자동 최적화하도록 공동 설계된 컴파일러
- 이 경로에서 프레임워크(JAX)를 선택하는 순간 컴파일러(XLA)가 결정되고, XLA의 최적 성능은 TPU에서만 발휘되어 HW 선택까지 연쇄 제약

  * Google이 이 수직통합 경로를 통해 TPU 생태계를 구축한 전략은 NVIDIA CUDA의 점진적 생태계 축적과 구별되는 '설계 단계의 종속(lock-in by Design)' 사례

- **(드라이버/런타임)** 상위 계층의 개방성과 무관하게 HW 종속을 최종 확정하는 '구조적 종속(Structural lock-in)'의 완성점
  - 프레임워크가 PyTorch(개방형)이고 컴파일러가 OpenXLA(오픈소스)더라도, 최종 연산은 반드시 칩 제조사의 폐쇄적 드라이버를 통과해야 실행가능
  - CUDA 드라이버는 비공개 소스로 배포되며, 자사 칩 아키텍처의 세밀한 변화에 맞춰 극도로 튜닝된 스케줄링·메모리 관리 로직을 내장
  - MIG·NVML처럼 HW-드라이버-운영툴까지 통합된 기능은 NVIDIA 스택에 깊게 결합되어 있어, 타사가 동일한 수준의 성숙도/생태계 호환성을 단기간에 따라잡기 어렵다는 평가[25]

- **(종합)** 1절에서 제시한 3유형 종속 프레임워크를 2~6절의 계층별 분석에 적용한 결과, 각 유형의 구체적 작동 양상과 중첩 구조가 확인
  - **(성능 종속)** 프레임워크 계층(2절: PyTorch CUDA 경로 성능 우위), 컴파일러 계층(3절: 연산 융합 범위·자동 튜닝 탐색 공간 격차로 동일 칩 최대 3배 처리량 차이), 가속 라이브러리 계층(4절: cuDNN 수천 개 커널 vs MIOpen 격차)에서 공통적으로 확인
  - **(설계 종속)** 2~3절에서 분석한 JAX → XLA → TPU 경로가 프레임워크 선택 컴파일러 확정 → HW 제약의 연쇄를 형성하며, 개별 계층의 성능 종속과 질적으로 상이한 교차 경로적 종속으로 확인
  - **(구조적 종속)** 5절에서 분석한 폐쇄적 드라이버가 상위 3개 계층의 개방성과 무관하게 HW 대체를 물리적으로 차단하며, MIG·NVML 등 운영 기능 독점이 전환 장벽을 강화하는 구조로 확인
  - 이 세 유형은 동일 생태계 내에서 중첩 가능하며, 중첩될 때 전환 비용이 기하급수적으로 증가
  - NVIDIA 생태계는 성능 종속(프레임워크~라이브러리), 구조적 종속(드라이버/런타임)을 동시에 갖추고, Google의 JAX-XLA-TPU 경로는 설계 종속을 추가로 구현한 점이 각각의 경쟁 우위

25 ABI Research(2025.05.28.), "NVIDIA's Strategy: Dominating AI Through Ecosystem, Access, and Interconnect"

○ 그러나 경쟁 HW가 충분히 큰 성능 우위나 가격 경쟁력을 보여줄 경우 기업들은 SW 전환 비용을 감수하고 새로운 플랫폼을 채택 가능

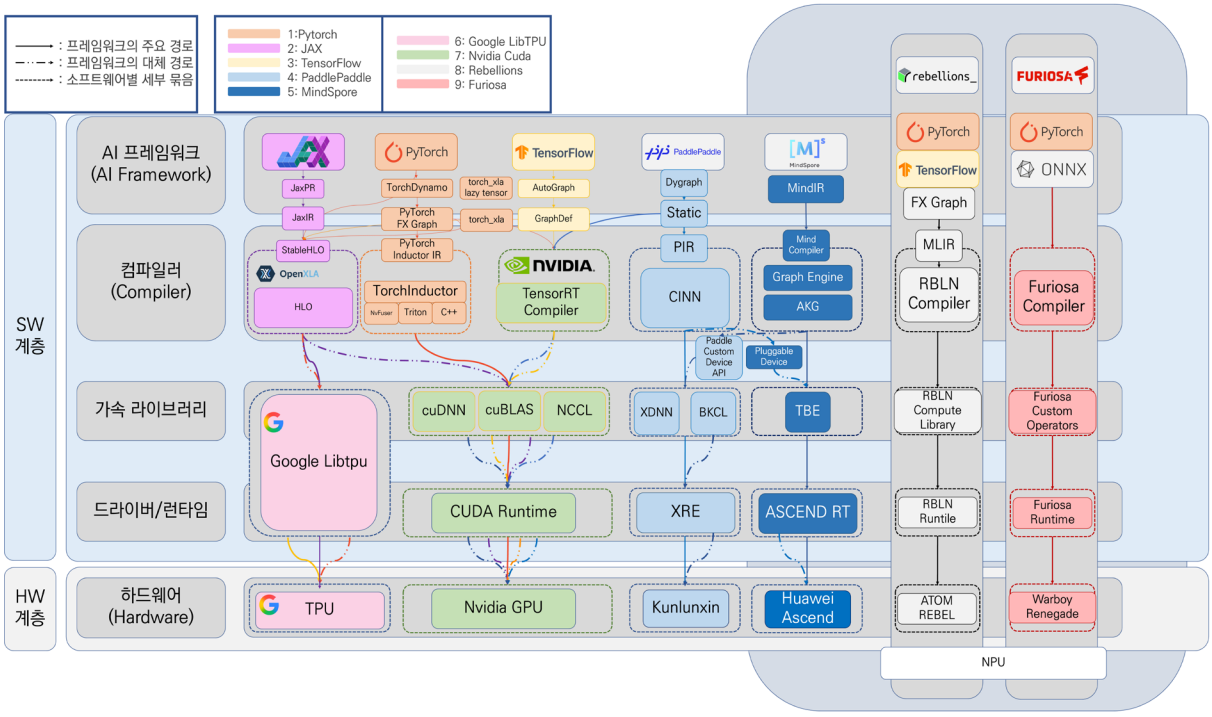
- AMD GPU들이 일부 워크로드에서 경쟁력 있는 성능을 보이면서 주요 기업들이 특정 벤더에 대한 의존도를 줄이려는 전략적 움직임

  * 2026년 2월 Meta는 AMD와 칩 구매 계약을 발표[26]

- AI 반도체 경쟁에서는 HW 성능과 SW 생태계가 모두 중요하며, 두 요소의 균형 있는 발전이 지속적 시장 경쟁력 확보의 핵심

  - HW 성능이 압도적으로 우수하면 SW 전환 비용을 상쇄할 수 있고, 성능이 비슷하면 생태계 완성도가 선택의 결정적 요인, 따라서 신규 AI 칩이 성공하려면 경쟁력 있는 HW 성능 확보와 함께 기존 생태계와의 호환성을 높이는 SW 투자 병행 필요

[그림 Ⅱ-3] 주요 HW별 SW 스택 정리



출처: 프레임워크, HW별 공식 문서 확인 후 SPRi 재구성

26 New York Times(2026.02.), "Racing to Catch Up With NVIDIA, AMD Signs Chips-for-Stock Deal With Meta"

Ⅲ 주요국·주요 기업의 AI SW 생태계 전략

○ II장에서 도출한 3유형 종속 프레임워크(성능·설계·구조적 종속)를 분석 틀로 활용하여, 주요 플레이어가 어떤 종속 메커니즘을 전략적으로 구축하거나 돌파하려 하는지를 검토

1. NVIDIA: 성능 종속 수직통합 생태계의 완성

○ NVIDIA의 수직통합 생태계 전략을 통한 가속기 시장 독점

- NVIDIA는 2006년 CUDA 플랫폼 출시 이후 약 20년간 HW-SW 수직통합 전략 구축 후 실행 중
  - Cusumano(2024)의 ACM Communications 분석에 따르면, NVIDIA는 5억 대 이상의 GPU 설치 기반과 CUDA 기반 애플리케이션 보유[27]
  - 데이터센터 GPU 시장에서 NVIDIA의 지배력은 단순한 HW 성능이 아닌 SW 생태계의 네트워크 효과에 기반
    * NVIDIA GPU가 많을수록 개발자들은 CUDA 기반 앱을 더 만들게 되고 CUDA 앱이 많아질수록 호환성을 위해 NVIDIA GPU를 구매하게 되어 네트워크 효과를 강화

- CUDA 생태계의 핵심 경쟁력은 개발자 커뮤니티 규모와 프레임워크 통합 수준에 있으며, 이는 경쟁사가 단기간에 복제하기 어려운 장벽을 형성
  - NVIDIA는 400만 명 이상의 등록 개발자와 수십만 개 기업이 사용하는 CUDA 기반 애플리케이션 생태계를 구축
  - PyTorch, TensorFlow 등 주요 AI 프레임워크가 CUDA에 최적화되어 있어, 개발자들이 NVIDIA GPU를 기본 선택지로 인식하는 경향이 강화
    * 이러한 현상을 '벤더 락인(Vendor lock-in)'으로 정의할 수 있으며, 전환 비용이 HW 성능 차이를 상쇄할 수 있음 (Cusumano, 2024)

27 Sze et al.(2024.03.), "Efficient Processing of Deep Neural Networks: A Tutorial and Survey"

- NVIDIA의 SW 스택은 cuDNN(딥러닝 연산), cuBLAS(선형대수), NCCL(분산 통신), TensorRT(추론 최적화) 등 계층별 라이브러리로 구성
  - 각 라이브러리는 NVIDIA GPU의 텐서 코어, NVLink 인터커넥트 등 HW 특성에 맞게 수년간 정밀 튜닝되어 최적 성능을 발휘
  - IEEE/ACM ISCA 2025에서 발표된 카이스트의 연구는 "NVIDIA의 강점은 CUDA 자체보다 풍부한 SW 생태계에 있다"고 주장[28]

    * 동 연구는 NPU 벤더가 PyTorch 등 고수준 프레임워크를 효과적으로 지원하면 CUDA가 절대적 장벽이 아닐 수 있다고 분석

- NVIDIA의 전략적 성공 요인은 HW 세대 간 SW 호환성 유지와 개발자 교육 투자를 통한 장기적 생태계 구축에 근간
  - 게이밍 GPU부터 데이터센터 가속기까지 동일한 CUDA 코드가 실행되는 통합 아키텍처 전략으로 개발자 투자가 전 제품군에서 활용 가능
  - 대학 교육 과정에 CUDA를 포함시키고, GTC 콘퍼런스와 개발자 프로그램을 통해 차세대 엔지니어를 자사 플랫폼에 훈련

- NVIDIA의 시장 지배력이 영구적이지는 않으며, HW 성능 격차가 충분히 클 경우 SW 전환이 일어날 수 있다는 분석도 존재

  * 모건스탠리는 2026년 반도체 전망에서 AI 학습 시장에서 NVIDIA의 지배력은 견고하지만, 추론 시장 같은 효율성이 중시되는 영역에서 대체제 도입이 검토되는 중[29]

  - Microsoft, Meta 등 주요 기업들이 AMD의 GPU 도입을 검토하고, 빅테크 기업들이 자체 AI 칩 개발을 추진하는 것은 이러한 가능성을 시사

- NVIDIA의 전략은, 프레임워크·컴파일러·라이브러리 계층의 최적화 격차를 통한 성능 종속과 폐쇄적 CUDA 드라이버를 통한 구조적 종속을 동시 활용하여, 어느 한쪽만 돌파해도 전환이 곤란한 이중 장벽을 형성한 구조

28 Lee et al.(2024), "Debunking the CUDA Myth Towards GPU-based AI Systems"

29 Investing.com(2024.05.), "Morgan Stanley upgrades SMIC on rising demand for China-made AI chips"

CUDA(Compute Unified Device Architecture) 개요

◆ 정의 및 개념

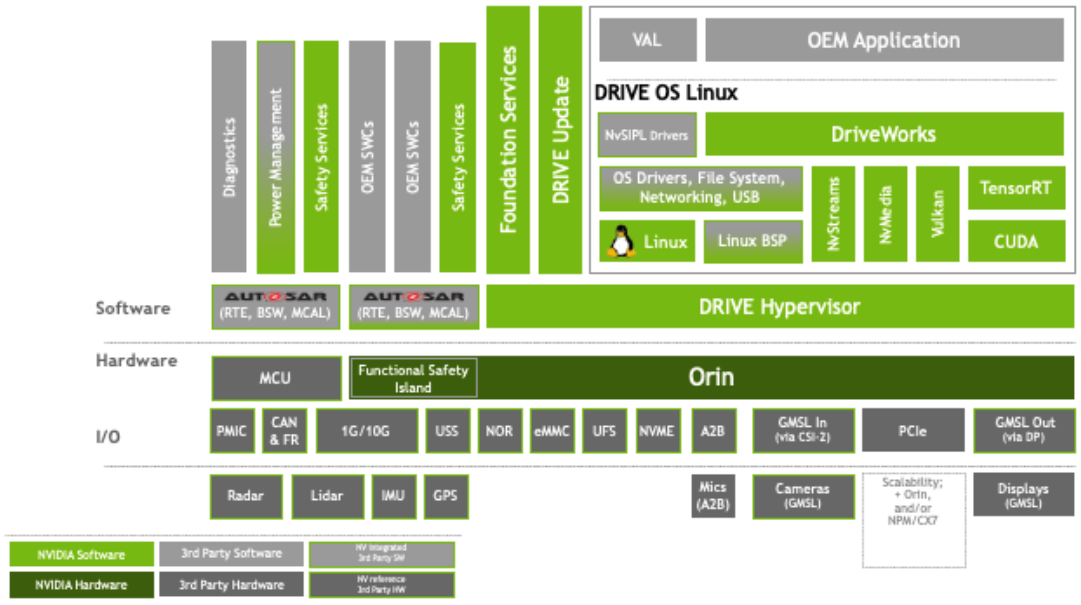
- CUDA는 NVIDIA가 2006년 출시한 병렬 컴퓨팅 플랫폼 및 프로그래밍 모델로, GPU를 그래픽 처리 외 범용 연산(GPGPU)에 활용할 수 있게 함
  * 그래픽 API에 대한 전문 지식이 필요한 기존 GPU 접근법에 비해, CUDA는 C/C++ 유사 문법을 제공, 일반 개발자의 접근성을 획기적으로 향상

◆ 핵심 구성요소

- **(CUDA Toolkit)** GPU 가속 애플리케이션 개발을 위한 컴파일러(nvcc), 디버깅 도구, 런타임 라이브러리를 포함한 통합 개발 환경

- **(CUDA-X 라이브러리)** AI 및 HPC 워크로드에 최적화된 GPU 가속 라이브러리 집합들로 딥러닝의 핵심 연산 최적화 라이브러리와 행렬계산에 도움
  - cuDNN: 컨볼루션, 어텐션 등 딥러닝 핵심 연산 최적화 라이브러리
  - cuBLAS: 행렬곱(GEMM) 등 선형대수 연산 가속 라이브러리
  - NCCL: 다중 GPU 간 초고속 통신을 위한 집합 통신 라이브러리
  - TensorRT: 추론 최적화 및 배포를 위한 고성능 런타임 엔진

[그림 III-1] CUDA SW 스택



2. Google: 설계 종속 프레임워크-컴파일러-칩 일체화 전략

○ Google은 TPU(Tensor Processing Unit) 개발과 함께 JAX/TensorFlow 프레임워크, XLA 컴파일러를 수직통합하여 독자적인 AI 컴퓨팅 스택을 구축

- TPU는 버전이 올라갈수록 추론 속도와 전력 효율이 계속 좋아졌으며 최신 TPU는 MLPerf 같은 공인 벤치마크에서도 비용 대비 성능이 지속 향상 중

  * 최근 세대 TPUV4는 v3 대비 2.1×성능, 2.7×성능/W를 달성했으며, v5e는 MLPerf 기준 LLM 추론 price-performance 2.7×(vs v4)를 보고[30]

- Google의 차별화된 전략은 '컴파일러 중심(Compiler-First)' 설계 철학으로, 수작업 커널 최적화 대신 XLA 컴파일러의 자동 최적화에 의존
  - XLA는 전체 프로그램 분석을 통해 연산 융합, 메모리 레이아웃 최적화 등을 자동 수행하여, 새로운 모델 구조에도 즉시 고성능 경로를 제공
  - Google Cloud 공식 문서에 따르면, 이 접근법은 "커널 중심 생태계가 새 연구를 따라잡기 위해 수작업 라이브러리를 기다려야 하는 병목을 해소"
  - NVIDIA의 cuDNN이 미리 최적화된 커널 라이브러리에 의존하는 것과 대조적인 설계 철학

- TPU와 XLA의 공동 설계는 HW-SW 최적화의 모범 사례로, 컴파일러가 칩의 메모리 접근 패턴을 예측 가능하게 만들어 효율을 극대화
  - 일반 GPU·CPU의 캐시, 비순차 실행, 프리페칭 등 동적 최적화는 평균 처리량에는 유리하나 지연시간 보장에는 불리
  - TPU는 예측 가능한 메모리 접근을 전제로 설계되어 캐시 없이도 높은 에너지 효율을 달성

    * Google은 TPU 기반 추론(서빙)에서 지연시간 SLO를 핵심 요구사항으로 제시하며, TPU는 설계 철학상 예측 가능한(결정론적) 실행이 꼬리 지연(p99) 관리에 유리하다는 분석 제시[31]

- Google은 XLA를 오픈소스 프로젝트 OpenXLA로 공개하여 업계 표준화를 추진하고, 다양한 HW 벤더의 참여를 유도하는 전략을 병행

30 Normal P. Jouppi et al.(2023), "TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings"

31 Google Cloud(2026), "Cloud TPU Inference"

- OpenXLA 프로젝트에는 Google 외에도 Alibaba, Amazon, AMD, Apple, Arm, Intel, Meta, NVIDIA 등이 참여하여 협력적 개발을 진행 중
- Google이 자체 TPU 생태계 강화와 동시에 AI 컴파일러 표준을 선점하여 장기적 영향력을 확보하려는 이중 전략으로 해석
- JAX 프레임워크는 함수형 프로그래밍 모델과 XLA의 긴밀한 통합으로 TPU에서 최고 성능을 발휘하도록 설계

• Google의 전략적 한계는 TPU가 Google Cloud 환경에서만 사용 가능하여 온프레미스(On-Premise)나 멀티클라우드(Multi Cloud) 환경에서 선택지가 제한
- 일부 고객들은 특정 클라우드 벤더에 대한 의존도 증가를 우려하여 TPU 채택을 주저하는 경향
- PyTorch/XLA 브릿지가 존재하나, CUDA 생태계 대비 라이브러리 지원과 커뮤니티 규모에서 아직 격차가 있다는 평가

• Google은 JAX-XLA-TPU 수직통합을 통한 설계 종속이라는 NVIDIA와 질적으로 상이한 경로를 구축하면서, OpenXLA 공개로 컴파일러 계층의 성능 종속 표준을 자사 주도로 재편하려는 이중 전략

3. 중국: 자립화와 생태계 구축의 이중 과제

○ 화웨이의 수직통합형 독자 생태계

• 화웨이는 미국의 수출 규제에 대응하여 Ascend NPU, CANN 컴퓨팅 아키텍처, MindSpore 프레임워크로 구성된 독자적 AI 스택을 구축
- 2019년 Ascend 910 출시와 함께 MindSpore 프레임워크를 공개하였으며, 2020 MindSpore를 오픈소스로 전환하여 생태계 확장을 추진
- 화웨이는 2019년 미국 제재 이후 “서방 기술 의존도를 줄이기 위한 전사적 노력”을 가속화
  * 2019년 HarmonyOS를 발표[32], 중국 내 자체 반도체 제조 인프라 구축[33]

32 Reuters(2024.06.), “Huawei’s Harmony Aims to End China’s Reliance on Windows, Android”
33 Financial Times(2025.05.), “Satellite Images Reveal Huawei’s Advanced Chip Production Line in China”

- CANN[34]은 화웨이의 CUDA 대응 SW 스택으로, Ascend NPU에서 AI 연산을 실행하기 위한 핵심 인프라
  - CANN은 고수준 AI 프레임워크와 Ascend 칩 사이의 브릿지 역할을 하며, 칩 수준 복잡성을 추상화하여 개발자 접근성을 높임
  - 2025년 8월 화웨이는 CANN을 완전 오픈소스로 공개하여 국내외 개발자 커뮤니티 확장과 생태계 활성화를 본격 추진[35]
  - TechRadar 분석에 따르면, CANN 오픈소스화는 "NVIDIA CUDA 독점에 대한 직접적 도전"으로 평가[36]

- 화웨이는 NVIDIA의 초기 CUDA 전략을 모방하여 주요 고객사에 엔지니어를 파견, CUDA 코드의 CANN 환경 이식을 직접 지원하는 전략을 채택
  - Financial Times 보도에 따르면, 화웨이는 Baidu, iFlytek, Tencent 등 중국 주요 AI 기업에 엔지니어 팀을 파견하여 코드 마이그레이션을 지원[37]
  - torch_npu 어댑터를 통해 PyTorch 모델이 Ascend NPU에서 실행될 수 있도록 하여 기존 개발자의 전환 장벽을 낮춤

- 화웨이 생태계의 현재 한계는 CUDA 대비 SW 성숙도와 개발자 경험 측면에서 격차가 존재
  - 업계 추정에 따르면, CANN 개발자 수는 CUDA 개발자에 비해 현저히 적으며, 동일 워크로드 실행 시 성능이 CUDA 대비 저성능[38]
  - DeepSeek 엔지니어들은 Ascend 910C가 H100 추론 성능의 약 60%를 달성할 수 있으나, CANN 최적화로 추가 향상이 가능하다고 보고[39]
  - CANN 8.0/9.0 버전에서 상당한 개선이 이루어졌으며, "CUDA-to-CANN" 자동 변환 도구 도입으로 마이그레이션 장벽이 낮아짐

- 미국 수출 규제가 오히려 중국의 SW 자립화를 가속화시키는 역설적 효과를 낳고 있다는 분석이 제기
  - CSIS 보고서는 "수출 규제가 중국의 기존 보조금 지원 개발 노력을 배가시키는 결과를 초래했다"고 평가[40]

---

34 Compute Architecture for Neural Networks

35 AI Times(2025.08.), "화웨이, AI 칩 소프트웨어 오픈소스 추진 … 엔비디아 CUDA에 도전"

36 TechRadar(2025.08.), "Brave or Foolhardy? Huawei Takes the Fight to NVIDIA CUDA by Making Its Ascend AI GPU Software Open Source"

37 Financial Times(2024.09), "Huawei's Bug-Ridden Software Hampers China's Efforts to Replace NVIDIA in AI"

38 TechNow(2025.11.), "Huawei CANN vs CUDA: Can Open-Source AI Break NVIDIA's Grip?"

39 Tom's Hardware(2025.02.), "DeepSeek research suggests Huawei's Ascend 910C delivers 60% of NVIDIA H100 inference Performance"

40 CSIS(2024.10.), "The Double-Edged Sword of Semiconductor Export Controls"

- ByteDance가 2027년까지 Ascend 칩에 56억 달러 규모의 주문을 확정했다는 보도는 중국 빅테크의 국산 칩 전환이 본격화됨을 시사[41]
- Merics 연구는 "중국이 AI의 모든 기술 계층에서 자립을 추구하고 있으며, 반도체 부문에 가장 강력한 국가 지원이 집중되고 있다"고 분석[42]

- 화웨이는 CANN(구조적 종속)과 MindSpore(성능 종속)를 갖춘 독자 스택으로, NVIDIA의 종속 구조를 자국 생태계 내에서 복제·내재화하려는 전략, 글로벌 개발자 규모 부족이 성능 종속의 실효성을 제약하는 구조적 한계

○ 바이두 PaddlePaddle의 종속 완화형 전략

- **(전략 기조)** 화웨이 MindSpore가 Ascend와의 설계 종속 경로를 구축한 것과 달리, PaddlePaddle은 복수 HW 백엔드 지원을 통해 특정 HW에 대한 성능 종속을 의도적으로 분산하는 전략 채택
  - PaddlePaddle은 NVIDIA GPU, AMD GPU, Intel CPU, 화웨이 Ascend, 다양한 국산 NPU 등 여러 HW 백엔드를 지원
  - 특정 HW 벤더에 종속되지 않고 중국 내 다양한 AI 칩 생태계와 협력하려는 전략으로 해석

- **(구조적 한계)** 해당 전략은 종속의 해소가 아닌 분산이라는 점에서 한계 존재
  - 화웨이 MindSpore가 Ascend와의 긴밀한 통합을 통한 최적 성능을 추구하는 반면, PaddlePaddle은 범용성과 개발자 접근성을 우선시
  - 복수 백엔드를 지원하되 각 백엔드별 최적화 깊이가 균일하지 않아, CUDA 경로에서의 성능이 여전히 가장 높을 경우 '형식적 개방성 ≠ 성능적 대등함'이라는 PyTorch와 동일한 성능 종속 구조 재현 가능성
  - Kunlun 칩과의 연동 시 PaddlePaddle → XTCL → XRE → Kunlun으로 이어지는 설계 종속 경로가 부분적으로 형성되나, CUDA 경로가 병존하여 JAX → XLA → TPU, MindSpore → CANN → Ascend 대비 설계 종속 강도 약함

- **(중국 AI SW 생태계의 종속 전략 분화)** 이러한 차이는 중국 AI SW 생태계가 종속 메커니즘의 관점에서 두 갈래로 분화하고 있음을 시사
  - **(화웨이 MindSpore)** 설계 종속 + 구조적 종속을 동시 구축하여 NVIDIA의 종속 구조를 자국 내에서 완결적으로 복제하는 전략 — 전환 비용의 극대화를 통해 Ascend 생태계 내 개발자를 고정

41 조선일보(2025.12.), "ByteDance Turns to Huawei AI Chips Amid U.S. Restrictions"
42 Merics(2025.07.), "China's Drive Toward Self-Reliance in Artificial Intelligence"

- **(Baidu PaddlePaddle)** 복수 HW 백엔드 지원으로 프레임워크 계층의 성능 종속을 분산하되, Kunlun 경로에서는 약한 수준의 설계 종속을 병행
- 화웨이가 '종속의 형성'을 통해 생태계를 고정하는 전략이라면, Baidu는 '종속의 완화'를 통해 생태계를 확장하는 전략으로, 동일한 탈NVIDIA 목표를 정반대의 종속 메커니즘으로 추구하는 구조

PaddlePaddle 개요

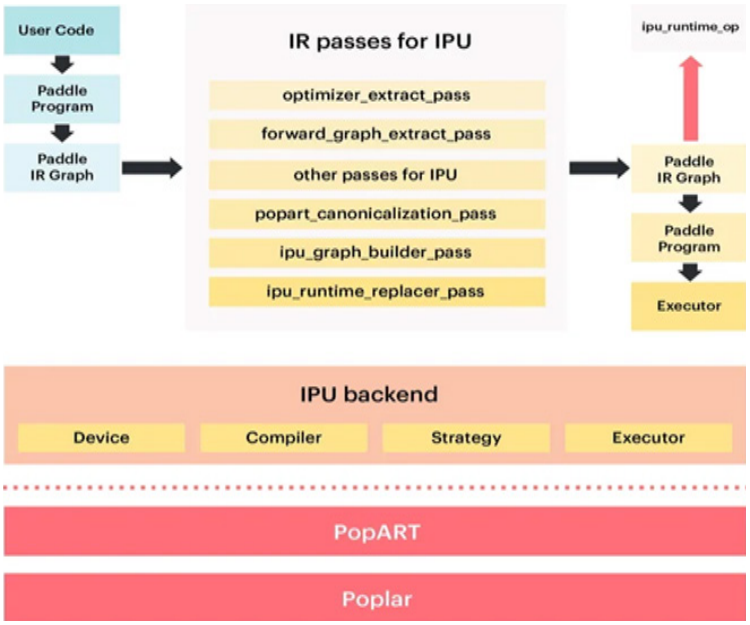
◆ 정의 및 개념

- PaddlePaddle(PArallel Distributed Deep LEarning)은 Baidu가 개발한 중국 최초의 독자 R&D 딥러닝 플랫폼으로, 2016년 오픈소스로 공개
  - Baidu 내부 운영을 위해 개발되었으며, UC Berkeley의 Caffe 프레임워크를 기반으로 RNN 지원을 추가하여 NLP 분야에서 초기 우위를 확보
  - '병렬 분산 딥러닝(Parallel Distributed Deep Learning)'의 약자로, 다중 GPU에서의 대규모 모델 학습 역량을 강조

◆ 특징

- **(산업용 응용 특화)** Tensor Flow, PyTorch가 연구·학계 중심인 반면, PaddlePaddle은 산업 현장 적용에 최적화된 설계 철학을 채택
- **(동적/정적 그래프 통합)** 연구 단계의 유연성(동적)과 배포 단계의 성능(정적)을 하나의 프레임워크에서 지원

[그림 III-2] PaddlePaddle 아키텍처



4. AMD·Intel: 후발 주자의 추격 전략

AMD ROCm의 CUDA 호환 전략

- AMD는 ROCm(Radeon Open Compute) 플랫폼을 통해 CUDA 대안 생태계 구축을 추진하나, SW 성숙도에서 NVIDIA와 상당한 격차 존재
  - ROCm은 2016년 출시된 오픈소스 GPU 컴퓨팅 플랫폼으로, MIOpen(cuDNN 대응), rocBLAS(cuBLAS 대응) 등의 라이브러리를 제공
  - The Register 분석에 따르면, HIPIFY 도구가 CUDA 코드를 ROCm용으로 자동 변환하나, "심각하고 생산 수준의 작업에는 부적합"하다는 의견[43]

- AMD의 핵심 전략은 CUDA 코드의 ROCm 이식을 최대한 용이하게 만들어 개발자 전환 장벽을 낮추는 것임
  - HIP[44]는 CUDA와 유사한 문법을 사용하여 개발자가 최소한의 코드 수정으로 AMD GPU를 타깃

- AMD의 한계는 라이브러리 최적화 수준과 프레임워크 통합 완성도가 CUDA에 미치지 못함
  - AMD GPU가 이론적 성능에서 우위임에도 실제 워크로드에서 NVIDIA가 앞선다고 지적[45]
  - 이 격차는 GPU 수가 증가할수록 더 벌어지는 경향이 있어, 대규모 분산 학습에서 AMD의 불리함이 더 두드러짐

Intel oneAPI의 범용성 추구

- Intel은 2019년 oneAPI 이니셔티브를 통해 CPU, GPU, FPGA, 가속기 등 다양한 HW를 단일 프로그래밍 모델로 지원하는 개방형 표준을 추진

- oneAPI는 SYCL 기반의 DPC++(Data Parallel C++) 언어를 핵심으로 하며, Khronos Group이 개발한 개방형 표준에 기반
  - Intel 공식 문서는 oneAPI를 "CUDA Lock-in에 대한 실행 가능한 대안"으로 포지셔닝하며, 벤더 중립성과 이식성을 핵심 가치로 제시[46]

---

43 The Register(2024.08.), "What's going on with AMD funding a CUDA translation layer, then nuking it"

44 Heterogeneous-Compute Interface for Portability

45 AIMultiple(2026.01.), "GPU Software for AI: CUDA vs. ROCm in 2026"

46 Intel(2024.06.), "oneAPI: A Viable Alternative To CUDA Lock-in"

- oneAPI의 전략적 차별점은 특정 HW에 종속되지 않는 크로스 플랫폼 호환성과 개방형 표준 기반의 생태계 구축
  - UXL(Unified Acceleration) Foundation이 Linux Foundation 산하에 설립되어 oneAPI 사양의 업계 표준화를 추진[47]
  - Codeplay 인수를 통해 oneAPI 코드가 NVIDIA 및 AMD GPU에서도 실행될 수 있는 플러그인을 제공[48]

- Intel의 한계는 AI 가속기 시장에서 GPU Max(Ponte Vecchio)의 경쟁력 부족과 Gaudi 가속기의 별도 SW 스택 문제
  - Gaudi 가속기는 oneAPI가 아닌 별도의 SynapseAI 스택을 사용하여, Intel 내부에서도 SW 생태계가 분절
  - The Register는 "PyTorch 지원이 HW 벤더마다 같은 의미가 아니며, Intel의 경우 맞춤형 PyTorch 버전이 필요하다"고 지적

AMD·Intel의 공통 한계

- CUDA의 성능 종속을 완화하는 전략(ROCm 호환, oneAPI 범용성)을 추구하나, 드라이버 계층의 구조적 종속까지 돌파하지는 못하는 구조
  - NVIDIA의 MIG·NVML 등 데이터센터 운영 기능에 대응하는 동등 수준의 런타임 인프라가 부재하여, 성능 격차를 좁히더라도 운영 전환 장벽이 잔존
  - 이는 성능 종속과 구조적 종속이 독립적 메커니즘이며, 한쪽만 완화해서는 전체 전환 비용을 충분히 낮출 수 없음을 보여주는 사례

5. 한국: 현황과 과제

산업 구조 재편: 합종연횡을 통한 규모의 경쟁력 확보

- 한국 AI 반도체 생태계는 리벨리온, 퓨리오사AI, 사피온 등 NPU 스타트업을 중심으로 형성되어 왔으나, 글로벌 경쟁 심화에 따라 산업 구조 재편이 본격화

47 Linux Foundation(2023.11.19.), "Unified Acceleration Foundation Forms to Drive Open Accelerated Compute and Cross-Platform Performance"

48 Intel(2022.06.01.), "Intel to Acquire Codeplay Software"

- 리벨리온과 SK텔레콤 계열사 사피온은 2024년 6월 합병을 발표하고 12월 통합법인을 공식 출범, 기업가치 약 1조 3,000억 원의 국내 최초 AI 반도체 유니콘 기업이 탄생[49]
- 합병을 통해 리벨리온은 SK텔레콤·SK하이닉스를 전략적 투자자로 확보하고, 미국·사우디 아라비아·일본 등 해외 시장 진출 기반을 마련
- 이번 합병은 개별 스타트업 규모로는 NVIDIA·Google 등 글로벌 기업과의 경쟁이 어렵다는 산업계의 공감대 아래 진행되었으며, 인력·자원·파트너십 면에서 글로벌 경쟁이 가능한 규모로의 전환을 의미

○ 리벨리온: PyTorch 생태계 주도와 풀스택 SW 전략

- 리벨리온은 ATOM 칩과 RBLN SDK를 중심으로, 국내 NPU 업체 중 가장 체계적인 SW 스택을 구축
  - RBLN SDK는 독자 컴파일러(프론트엔드/백엔드 2단계 구조), 연산 라이브러리(Compute Library), 런타임, 드라이버, 펌웨어로 구성된 풀스택 SW로, 본 보고서의 4계층 분류에 대응하는 완결된 구조
  - PyTorch 2.0 네이티브 지원과 기존 PyTorch 코드를 최소한의 수정으로 NPU에서 실행 가능하며, TensorFlow 및 Hugging Face 모델을 포함한 200개 이상의 레퍼런스 모델을 지원
  - RBLN 컴파일러의 프론트엔드는 딥러닝 모델을 중간 표현(IR)으로 추상화하고, 백엔드가 HW 특성에 맞춰 최적화된 명령 스트림과 프로그램 바이너리를 생성하는 구조로 컴파일러 계층의 역할을 자체 구현

- 차세대 칩 로드맵 측면에서 리벨리온은 LLM 가속에 특화된 REBEL 칩을 삼성 4nm 공정으로 개발 중이며, UCIe-Advanced 기반 칩렛(Chiplet) 기술을 적용한 REBEL-Quad까지 계획
  - Hot Chips 2025에서 REBEL-Quad를 공개하였으며, 144GB HBM3E 메모리와 4.8TB/s 대역폭을 탑재하여 수백억 파라미터 규모 모델을 단일 칩에서 처리하는 것을 목표[50]

○ 퓨리오사AI: 전력 효율 특화와 글로벌 빅테크 검증

- 퓨리오사AI는 텐서 수축 프로세서(TCP) 아키텍처 기반의 RNGD 칩으로 AI 추론 시장에서 전력 효율 중심의 차별화 전략을 추진

---

49 비즈조선(2024.12.02.), "리벨리온·사피온, 합병 완료 … 기업가치 1조 3,000억 원"

50 Rebellions(2025.08.27.), "Rebellions Debuts REBEL-Quad at Hot Chips 2025, Breaking AI's Energy Tax with High-Performance Chiplet Innovation"

- RNGD는 TSMC 5nm 공정으로 제조되며, FP8 기준 512 TFLOPS 성능을 TDP 180W로 달성하여, GPU 칩 대비 와트당 성능이 2배 이상으로 평가[51]
- LG AI연구원의 EXAONE 3.5 32B 모델 테스트에서 RNGD 4장 단일 서버로 4K 컨텍스트에서 60 tokens/s, 32K 컨텍스트에서 50 tokens/s를 달성, 동일 전력 조건에서 RNGD 기반 랙이 GPU 기반 랙 대비 EXAONE 모델에서 3.75배 더 많은 토큰을 생성할 수 있었다고 보고

• 퓨리오사AI의 SW 스택(Furiosa SDK)은 PyTorch JIT 컴파일러 네이티브 지원, vLLM 호환 서빙 프레임워크(Furiosa LLM), OpenAI 호환 API, Kubernetes 통합을 제공
- Furiosa LLM은 오픈소스 vLLM 프레임워크의 드롭인 대체(Drop-in replacement)를 목표로 설계되어 기존 GPU 기반 서빙 인프라에서의 전환 장벽을 최소화
- NXT RNGD 서버는 Furiosa SDK와 런타임이 프리인스톨(Pre-install)된 턴키 시스템으로 3kW 시스템 전력에서 4 PFLOPS(FP8) 성능을 제공

• 글로벌 빅테크의 기술 검증이 퓨리오사AI의 시장 신뢰도를 뒷받침
- 오픈AI코리아 개소식에서 RNGD 2장으로 gpt-oss 120B 모델 기반 챗봇을 실시간 시연하며 글로벌 호환성을 입증
- Microsoft Azure Marketplace에 RNGD를 공식 출시하여 클라우드 기반 접근성을 확보하고, 2026년 TSMC 양산 4,000장을 인도받아 본격 양산

○ 추론 특화 경쟁력: MLPerf 벤치마크를 통한 글로벌 기술력 입증

• 국내 NPU 업체들은 AI 추론 영역에서 글로벌 빅테크와 대등한, 일부 영역에서는 이를 상회하는 성능을 MLPerf 벤치마크를 통해 객관적으로 입증
- 리벨리온 ATOM은 MLPerf v3.0에서 언어모델(BERT-Large) 처리 시간 4.3ms를 기록, NVIDIA A2(6.09ms) 대비 1.4배, 퀄컴 Cloud AI 100(7.55ms) 대비 1.75배 빠른 속도를 달성
- 비전 모델(ResNet50)에서도 싱글스트림 0.239ms를 기록하여 NVIDIA T4(0.82ms) 대비 3.4배, 퀄컴 AI 100(0.34ms) 대비 1.4배 우수한 성능을 시현
- 퓨리오사AI Warboy는 2021년 MLPerf에서 대규모 이미지·비디오 테스트에서 NVIDIA 동급 칩을 능가하는 성능을 시연하며 글로벌 시장의 주목을 받았고, 사피온 X220은 2022년 테스트에서 2.2배 높은 전력 효율

51 FuriosaAI(2025.07.22.), “LG AI Research Taps FuriosaAI to Achieve 2.25x Better LLM Inference Performance vs. GPUs”

- 이러한 추론 특화 경쟁력은 현재 AI 시장의 구조적 변화와 맞물려 전략적 의미가 큼
  - AI 학습 시장에서 NVIDIA의 지배력이 견고한 반면, 추론 시장에서는 효율성이 중시되어 대체제 도입이 본격 검토되고 있는 중
    * 가트너는 추론 시장이 갈수록 커지고 있으며 GPU는 물론 ASIC, TPU 등 다양한 AI 가속기 기술에 대한 투자가 확대되고 있다고 설명
  - 전 세계 AI 워크로드에서 추론이 차지하는 비중이 학습 대비 빠르게 증가하고 있어, 추론 영역의 전력 효율과 비용 경쟁력은 데이터센터 운영자에게 점점 핵심적인 선택 기준으로 부상
  - 국내 NPU들이 공통적으로 보여주는 '높은 추론 성능 + 낮은 전력 소비' 조합은 AI 인프라의 전력·비용 문제가 심화되는 글로벌 시장에서 차별화된 가치를 제공할 수 있는 포지션

- 중장기적으로는 추론 영역의 강점을 기반으로 학습 영역까지 역량을 확장하는 로드맵이 필요하며, 리벨리온의 차세대 칩 REBEL이 LLM 학습 가속을 목표로 개발되고 있는 점은 이러한 방향성을 반영

○ 한국 NPU 생태계의 구조적 과제

- 3유형 종속 관점에서 국내 NPU의 현재 포지션은 PyTorch 네이티브 지원으로 프레임워크 계층의 성능 종속 진입장벽을 우회하는 데 성공

- 컴파일러·라이브러리 계층의 최적화 깊이(커널 수, 자동 튜닝 범위)에서 CUDA 대비 격차가 잔존하여 성능 종속의 완전한 해소에는 미도달
  - 드라이버/런타임 계층에서는 자체 SDK를 통해 구조적 종속의 영향을 받지 않는 독립 경로를 확보하였으나, 이 독립성이 반대로 NVIDIA 생태계와의 호환성 부재로 작용하여 전환 유인을 약화시키는 역설 구조

- HW 성능 입증에 비해 SW 생태계의 깊이와 폭에서 NVIDIA CUDA와의 격차가 여전히 존재
  - RBLN SDK가 200개 이상의 레퍼런스 모델을 지원하나, CUDA 생태계의 수천 개 최적화 라이브러리와 수십만 개 애플리케이션 규모와는 차이
  - 국내 NPU의 강점이 추론 영역에 집중되어 있어, 학습(training) 영역에서의 HW·SW 역량 확보가 중장기 과제로 남아있음

- 개별 업체가 각자의 SDK·컴파일러·런타임을 독립적으로 개발하는 구조는 제한된 인력과 자원의 분산을 초래

- 컴파일러 IR 계층이나 런타임 인터페이스에서의 공통 표준화가 이루어지지 않으면 개발자는 칩마다 별도의 학습 비용을 부담해야 하며, 이는 전체 생태계의 확장을 저해
- OpenXLA, MLIR 등 글로벌 오픈소스 프로젝트에 대한 참여와 기여를 통해 국제 표준 형성 과정에서의 발언권을 확보하고 국내 NPU 간 공통 계층을 모색할 필요

• AI 컴파일러·시스템 SW 전문 인력의 절대적 부족이 SW 생태계 구축의 가장 근본적인 병목
- 리벨리온이 PyTorch 교육 과정을 대학에 도입하고 있으나 컴파일러 설계·디바이스 드라이버 최적화 등 저수준 시스템 SW 인력 양성은 산학연 차원의 체계적 투자가 필요한 장기 과제

[표 Ⅲ-1] 주요국·기업의 AI SW 생태계 전략 비교

구분	NVIDIA	Google	화웨이	AMD	Intel	한국 NPU
종속 전략 유형	성능 종속 +구조적 종속 동시 활용	설계 종속 구축 +성능 종속 표준화 병행	성능·설계· 구조적 종속 자국 내 복제	성능 종속 분산 +약한 설계 종속 병행	성능 종속 완화 시도	성능 종속 완화 시도
종속 형성/ 돌파 방향	종속 형성 (이중 장벽)	종속 형성 (별도 경로)	종속 복제 (자국 내 재현)	종속 완화 (생태계 확장)	종속 돌파 (호환 경로)	종속 돌파 (표준 경로)
핵심 SW 스택	CUDA, cuDNN, TensorRT, NCCL	XLA, JAX, libtpu	CANN, MindSpore, AscendCL	PaddlePaddle, XTCL, XRE	ROCm, HIP, MIOpen	oneAPI, SYCL, oneDNN
드라이버 개방성	비공개(구조적 종속 핵심)	비공개 (libtpu)	비공개(NVIDIA 구조 복제)	비공개 (Kunlun)	오픈소스 (amdgpu)	오픈소스 (i915/xe)
데이터센터 관리	MIG·NVML (독점 장벽)	Cloud TPU 관리(내부)	MindX·VNPU (MIG 대응)	기본 모니터링 (초기)	ROCm SMI (부분 대응)	XPU Manager (부분 대응)
PyTorch 통합 방식	네이티브 (CUDA 백엔드)	XLA 경유 (설계 종속 경로)	torch_npu (호환 계층)	자체 프레임워크 우선+PyTorch 지원	네이티브 (HIP 백엔드)	IPEX 별도 확장
생태계 성숙도	최고	높음	성장 중 (중국 내 선도)	성장 중 (중국 2위)	중간	중간
개발자 규모	400만+	비공개 (GCP 내부 중심)	성장 중 (중국 내)	800만 +(공식 발표)	제한적	제한적
종속 구조의 핵심 강점	최적화 격차(성능) +폐쇄적 드라이버 (구조적)의 이중 장벽	SW 선택이 곧 HW를 확정하는 설계 경로 고정	3유형 완결적 복제에 의한 자국 내 전환 장벽	복수 HW 지원으로 특정 벤더 종속 회피	CUDA 호환으로 전환 비용 저감	개방형 표준으로 HW 중립성 확보
종속 구조의 핵심 한계	개방형 표준·오픈소스의 장기적 격차 축소 압력	TPU 외부 확장성 제약, Cloud 종속	글로벌 개발자 규모 부족으로 성능 종속 실효성 제약	백엔드별 최적화 격차로 CUDA 경로 성능 종속 재현 가능성	구조적 종속 (드라이버 운영 기능) 돌파 미달	구조적 종속 (드라이버 운영 기능) 돌파 미달

Ⅳ 결론 및 정책적 시사점

1. 연구 결과 요약

○ AI 인프라 경쟁력은 단순 HW 사양만으로 결정되지 않으며, SW 스택의 성숙도가 HW의 실제 사양과 종속성을 결정

* AI SW 스택은 프레임워크 → 컴파일러 → 가속 라이브러리 → 드라이버/런타임으로 구성되며 HW의 성능과 종속성을 결정

- 4개 계층 각각이 AI 칩의 실질적 경쟁력 형성에 핵심적 역할을 수행하며 어느 한 계층이 미흡해도 전체 성능이 저하됨

  * HW의 이론적 대역폭을 실현하려면 SW가 비동기 연산 조율과 통신 병목 회피를 수행해야 함[52]

- 4계층의 종속 메커니즘을 유형화한 결과, 세 유형의 종속이 구분되며 이들이 중첩될 때 전환 비용은 기하급수적으로 증가
  - **(성능 종속)** 상위 3개 계층(프레임워크·컴파일러·가속 라이브러리)에서 최적화 격차에 기반하여 대안이 존재하나 사실상 특정 HW로 수렴하는 구조. SW 투자로 격차 축소가 가능한 유형
  - **(설계 종속)** 프레임워크-컴파일러-HW가 설계 단계에서 공동 최적화되어, 특정 SW 선택이 곧 HW 경로를 확정하는 구조. JAX → XLA → TPU가 대표 사례이며, 성능 종속보다 전환 비용이 높음
  - **(구조적 종속)** 드라이버/런타임 계층의 폐쇄적 구조가 HW 대체를 물리적으로 차단하는 구조, 상위 계층이 모두 개방형이더라도 최종 연산은 칩 제조사의 독점 드라이버를 통과해야 실행 가능

    * 성능 종속은 SW 투자로 완화 가능하나, 구조적 종속은 독자 드라이버 생태계 확보 없이는 돌파 곤란

- 주요국·기업 사례 분석 결과, 성공적 AI 생태계 구축에는 프레임워크 호환성 + 개발자 경험 + 10년 이상 지속 투자가 공통 요소임
  - NVIDIA: 2006년 CUDA 출시 이후 20년간 400만 개발자 커뮤니티, 수천 개 CUDA 앱 생태계 축적
  - 화웨이: 2019년 미국 제재 이후 Ascend-CANN-MindSpore 독자 스택 구축, 2025년 CANN 오픈소스화

52 Liu et al.(2024), “Aceso: Efficient Parallel DNN Training through Iterative Bottleneck Alleviation”

2. K-NPU 포지션 진단

○ 현재 한국 NPU 생태계는 '프레임워크 계층 진입에는 성공했으나, 컴파일러·라이브러리 계층의 성능 격차와 운영 생태계의 규모 부족이 시장 확산을 제약하는' 구조적 위치

- **(성능 종속: 부분 돌파)** 프레임워크 계층의 진입장벽 우회에는 성공하였으나, 컴파일러·라이브러리 계층의 최적화 깊이에서 CUDA 대비 격차가 잔존
  - 리벨리온·퓨리오사AI 모두 PyTorch 네이티브 지원으로 개발자가 최소한의 코드 수정으로 NPU를 타깃할 수 있는 환경을 확보
  - vLLM 백엔드 통합(리벨리온), Furiosa LLM 드롭인 대체(퓨리오사AI) 등 추론 서빙 생태계 진입에도 성공
  - 그러나 RBLN SDK의 200개 레퍼런스 모델 대비 CUDA 생태계의 수천 개 최적화 커널·수십만 개 애플리케이션과의 규모 격차가 현저
  - 컴파일러의 자동 튜닝 탐색 공간, 가속 라이브러리의 커널 수·정밀도별 최적화 경로에서 성능 종속이 완전히 해소되지 않은 상태

- **(설계 종속: 해당 없음)** K-NPU는 특정 프레임워크-컴파일러-HW 수직통합 경로를 구축하지 않았으며 PyTorch·ONNX 등 개방형 경로를 통한 생태계 진입 전략을 채택
  - 이는 Google의 JAX → XLA → TPU나 화웨이의 MindSpore → CANN → Ascend과 달리 설계 종속에 의한 개발자 고정 효과를 갖지 못한다는 의미
  - 반면, 특정 수직통합 경로에 갇히지 않아 글로벌 생태계 변화(OpenXLA, Triton 확장 등)에 유연하게 대응 가능한 구조적 이점도 존재

- **(구조적 종속: 독립 확보, 역설 존재)** 자체 SDK·드라이버를 통해 NVIDIA의 폐쇄적 드라이버로부터 독립된 경로를 확보하였으나 이 독립성이 역설적으로 전환 유인을 약화
  - NVIDIA의 CUDA 드라이버·MIG·NVML에 종속되지 않는 독자 런타임을 보유하여 구조적 종속의 영향을 받지 않는 독립 경로를 확보
  - 그러나 이 독립성은 동시에 NVIDIA 생태계와의 호환성 부재를 의미하며 기존 CUDA 환경에서 운영 중인 데이터센터가 NPU로 전환할 때 운영 도구(모니터링·가상화·스케줄링) 전체를 재구축해야 하는 부담으로 작용
  - MIG 동등의 GPU 자원 분할 기능, NVML 수준의 데이터센터 관리 도구가 부재하여 성능 격차를 좁히더라도 운영 수준의 전환 장벽이 잔존하는 구조

- **(종합 진단)** K-NPU의 현재 포지션은 '프레임워크 계층 진입 성공 → 컴파일러·라이브러리 계층 성능 격차 잔존 → 운영 생태계 규모 부족'의 3단계 과제가 순차적으로 존재
  - 1단계(프레임워크 진입)는 달성, 2단계(성능 종속 해소)는 진행 중, 3단계(운영 생태계 확보)는 초기 단계
  - 각 단계의 과제는 독립적이 아니라 연쇄적으로 작동하며, 2단계의 성능 격차가 좁혀지지 않으면 3단계의 운영 레퍼런스 축적이 곤란하고, 3단계의 레퍼런스가 없으면 2단계의 투자 정당성 확보가 어려운 순환 구조
  - 따라서 정책 개입은 이 순환 구조를 깨는 방향, 즉 2단계와 3단계를 동시에 추진하여 선순환을 유도하는 데 초점을 맞출 필요

3. 정책적 시사점

○ (성능 종속 해소) 컴파일러·라이브러리 계층의 최적화 깊이 확보

- **(현안)** K-NPU가 프레임워크 계층 진입에는 성공했으나, 컴파일러·가속 라이브러리 계층에서 CUDA 대비 커널 수·자동 튜닝 범위·정밀도별 최적화 경로의 격차가 잔존하며, 이 격차가 동일 모델에서의 처리량 차이로 직결
  - 동일한 H100 칩에서도 SW 최적화 수준에 따라 처리량이 3배 이상 차이, 칩 확보만으로는 AI 인프라 경쟁력을 확보할 수 없음을 시사
  - 현재 국가 R&D의 AI 반도체 투자는 칩 설계에 집중되어 있으나, 칩의 잠재력을 실제 가치로 전환하는 것은 결국 SW

- **(제안)** 칩 설계 중심의 기존 R&D 지원을 컴파일러·런타임·SDK 등 SW 생태계 전반으로 확대하는 HW-SW 균형 발전 패러다임 전환
  - 정부 R&D 예산에서 컴파일러·런타임·SDK 등 SW 개발 비중을 확대하고, HW 사업에 SW 공동 설계를 필수 요건으로 제도화
  - 데이터-모델-서비스를 아우르는 참조 아키텍처(Reference Stack)를 정의하고, 공공·민간에서 재사용 가능한 표준형 데이터 파이프라인·MLOps·모델 서빙 환경 개발을 지원
  - 정부 과제 평가 시 단순 연산 성능 수치보다 표준 프레임워크와의 SW 호환성 및 실질적인 런타임 최적화 역량을 주요 지표로 반영

- **(기대효과)** 이를 통해 국내 연구기관, 스타트업, 중견기업이 동일한 SW 기반 위에서 실험과 배포 및 확장을 수행할 수 있도록 함으로써, AI 서비스 개발의 고정비를 낮추고 혁신 속도를 높이는 효과 창출

(성능 종속 완화) 글로벌 오픈소스 생태계 참여를 통한 최적화 격차 축소

- **(현안)** 성능 종속의 본질은 '최적화 깊이의 비대칭'이며, 이를 개별 NPU 업체가 독자적으로 해소하는 것은 자원 제약상 비현실적
  - OpenXLA, MLIR, Triton 등 컴파일러 계층의 오픈소스 프로젝트는 HW 종속을 완화하는 핵심 수단이며, 이러한 프로젝트에 대한 기여가 곧 국제 생태계에서의 발언권으로 연결
  - 개별 업체가 각자의 SDK·컴파일러·런타임을 독립적으로 개발하는 구조는 제한된 인력과 자원의 분산을 초래
  - 컴파일러 IR 계층이나 런타임 인터페이스에서의 공통 표준화가 이루어지지 않으면 개발자는 칩마다 별도의 학습 비용을 부담해야 하며, 이는 전체 생태계의 확장을 저해

- **(제안)** 글로벌 AI 개발자 대다수가 사용하는 PyTorch 호환성 확보를 최우선으로 하되, OpenXLA·MLIR 등 컴파일러 표준 프로젝트 참여를 통해 국제 생태계 연결고리를 확보
  - 국가 R&D 및 공공사업에서 개발된 AI·SW 성과는 기본적으로 오픈소스 혹은 개방 API 형태로 공개
  - 핵심 오픈소스 프로젝트(딥러닝 프레임워크, 모델 서빙, MLOps, 데이터 품질관리 등)를 발굴하고 장기적으로 지원
  - 국내 NPU 간 컴파일러 IR 계층이나 런타임 인터페이스에서의 공통 표준화를 모색하여 개발자의 칩별 학습 비용을 절감

- **(기대효과)** 단순한 비용 절감을 넘어, 국내 개발자·연구자의 역량을 글로벌 수준의 개발 문화와 연결하고, 장기적으로는 국내 기업이 글로벌 AI·SW 생태계의 핵심 기여자로 자리 잡는 기반 구축

(구조적 종속 우회) 공공부문 선도 수요를 통한 운영 레퍼런스 확보

- **(현안)** K-NPU는 NVIDIA의 구조적 종속으로부터 독립된 경로를 확보하였으나, 그 독립성이 오히려 기존 CUDA 생태계 운영 환경과의 호환성 부재로 작용하여 전환 유인을 약화시키는 역설
  - 신규 AI 칩과 자체 SW 스택이 시장 신뢰를 얻기 위해서는 대형 데이터센터 단위의 실제 워크로드 운영 실적이 필수적

- 하지만 검증되지 않은 인프라로의 전환에 따른 SW 마이그레이션 비용과 리스크를 민간이 감수하기 어려운 구조
- 퓨리오사AI·리벨리온 등 국내 NPU가 글로벌 수준의 추론 성능을 입증했으나, 대규모 상용 환경에서의 장기 운영 레퍼런스는 아직 초기 단계
- 이 순환 구조(레퍼런스 부족 → 도입 주저 → 레퍼런스 축적 불가)를 민간 자율에만 맡기면 해소가 곤란

• **(제안)** 국가 AI 데이터센터 구축 및 공공 클라우드 전환 사업에 국산 NPU와 전용 SW 스택 도입을 장려하여, 초기 생태계의 자생력을 높이고 대규모 실증 기회를 제공
- 공공·민간의 대표 프로젝트를 활용하여 AI 시스템 개발 및 운영에 관한 모범 사례(Best Practice)를 축적하고 공유
- 민간이 겪는 SW 마이그레이션 비용과 오류 리스크를 정부 시범 사업이 일부 분담하여 실질적 도입 장벽을 완화
- 특히 NPU의 전력 효율 우위가 부각되는 AI 추론 워크로드를 중심으로 공공부문 적용을 우선 추진하여, 학습 영역의 NVIDIA 지배력과 직접 경쟁하지 않는 전략적 진입점 확보

• **(기대효과)** 공공부문이 '첫 번째 고객(First Mover)' 역할을 수행함으로써 민간 시장의 채택을 유도하는 레퍼런스 효과를 창출하고, 운영 생태계의 규모를 확보하여 구조적 종속 우회 경로를 실증

○ (전환 비용 가시화) TCO 기반 평가 체계 도입

• **(현안)** HW 대체는 '전환 비용'이라는 공통 메커니즘을 통해 억제되며, 현재 AI 인프라 도입 의사결정이 HW 단가와 이론적 연산 성능(FLOPS) 중심으로 이루어지고 있어 SW·운영·전환 비용이 체계적으로 평가되지 않는 구조

* AI 인프라의 HW 구입비(CapEx)는 전체 TCO의 약 절반에 불과하며, 나머지 절반은 운영 기간 중 누적되는 전력·인력·SW 비용으로 구성[53]

- 기존 GPU 인프라에서 NPU로의 전환 시 HW 비용 외에도 모델 포팅, 컴파일러 호환성 확보, 서빙 SW 재구성, 개발자 재교육 등 SW 전환 비용이 상당 규모로 발생하나, 이를 정량화하는 표준 방법론이 부재

53 Hyperion Research, Intel, Ansys(2024.08.), "Understanding the Total Cost of Ownership in HPC and AI Systems"

- **(제안)** 국가 AI 데이터센터 구축 및 공공 클라우드 전환 사업의 HW 선정 시, HW 단가·FLOPS 중심 평가에서 TCO 기반 종합 평가로 전환
  - 공공 조달 평가 기준에 '5년 TCO 분석서' 제출을 의무화하되, HW 비용·전력비·SW 전환 비용·인력비·운영 리스크를 포괄하는 표준 TCO 산출 양식을 정부 차원에서 마련
  - NPU의 전력 효율 우위가 TCO에 미치는 효과를 시뮬레이션할 수 있는 공개형 TCO 비교 도구를 배포해 민간 기업의 합리적 인프라 투자 유도

- **(기대효과)** TCO 기반 평가 체계가 도입되면 HW 단가 경쟁에서 불리한 국산 NPU가 전력 효율·운영비 측면의 구조적 강점을 정량적으로 입증할 수 있는 경쟁 환경이 조성되고 동시에 SW 생태계 투자의 경제적 당위성이 부각

(생태계 기반 역량) AI-SW 통합적 인재 양성과 조직 역량 강화

- **(현안)** 성능 종속 해소와 오픈소스 생태계 참여의 실행 주체는 결국 AI 컴파일러·시스템 SW 전문 인력이며 인력의 부족이 정책의 가장 근본적인 병목
  - 컴파일러·런타임·드라이버 계층은 반도체 공학과 컴퓨터 SW 공학의 경계에 위치한 고도의 전문 영역
    * NVIDIA가 대학 교육 과정에 CUDA를 포함시켜 차세대 개발자를 자사 플랫폼에 훈련시킨 전략은, 인재 양성이 곧 생태계 종속의 시작점임을 보여주는 사례[54]

- **(제안)** SW·AI 전문 인력 양성을 위한 교육 과정을 "AI 시스템 엔지니어링" 관점에서 통합적으로 재설계하고, 산업 측면에서는 기존 SW 개발 조직에 AI 전담 플랫폼의 내재화를 촉진하는 정책 필요
  - 기존 SI 및 SW 개발 조직에 AI 개발 역량의 내재화를 위해 재직자 대상 AI-SW 통합 재교육 프로그램으로 개발자 역량 강화
  - 리벨리온이 PyTorch Foundation 멤버로 참여하며, 국내 대학(연세대 한양대·UST 등)과 PyTorch 관련 학부/대학원 교육을 협력하고, PyTorch-NPU 네이티브 연동 생태계 활동(MODULABS 등)을 병행[55]

- **(기대효과)** 기업의 AI 모델 개발 능력과 더불어 이를 SW 기반의 제품 및 서비스로 구현·확산할 수 있는 건실한 SW 생태계를 위한 인적 역량 확보

54 NVIDIA(2011.05.10.), "Universities and Research Institutions Around the World Embrace Parallel Computing Using GPUs"
55 PyTorch(2024.12.21.), "Rebellions Joins the PyTorch Foundation as a General Member"

SW 경쟁력이 곧 AI 경쟁력

- 보고서가 분석한 바와 같이 AI 인프라의 경쟁은 칩 성능이 아닌 SW 생태계의 종속 구조에서 결정되며 정책의 무게 중심도 이에 맞춰 이동해야 함

- 성능 종속은 컴파일러·라이브러리 투자와 오픈소스 참여로 격차를 축소할 수 있고, 설계 종속은 개방형 경로를 유지함으로써 회피할 수 있으며, 구조적 종속은 공공부문 실증·테스트베드 확보로 운영 레퍼런스를 확보하여 우회

- 세 방향의 정책이 동시에 추진될 때, 2단계(성능 격차 해소)와 3단계(운영 생태계 확보)의 순환 구조가 깨지고, K-NPU 생태계의 자생적 성장 가능

- AI 강국은 건실한 SW 생태계 없이는 불가능하다는 점을 인식하고, 상보관계에 있는 SW와 AI의 동반 성장을 지원하는 지속적인 정책 추진이 필요

참고문헌

1. 국내문헌

- AI Times(2025.08.), "화웨이, AI 칩 소프트웨어 오픈소스 추진 … 엔비디아 CUDA에 도전"
- 아주경제(2026.01.), "퓨리오사AI, 2세대 AI 칩 'RNGD' 첫 양산"
- 바이라인네트워크(2023.04.), "MLPerf 벤치마크서 퀄컴·엔비디아 제친 리벨리온"
- 비즈조선(2024.12.02.), "리벨리온·사피온, 합병 완료 … 기업가치 1조 3,000억 원"
- 조선일보(2025.12.), "ByteDance Turns to Huawei AI Chips Amid U.S. Restrictions"
- CIO Korea(2025.07.), "AI 반도체 스타트업 퓨리오사AI, 기업고객 겨냥해 LG와 협력체결"
- ETRI(2025.10.), "온디바이스 AI를 위한 시스템 소프트웨어 기술 동향"
- FuriosaAI(2025.07.22.), "LG AI Research Taps FuriosaAI to Achieve 2.25x Better LLM Inference Performance vs. GPUs"
- FuriosaAI(2025.10.), "Introducing Furiosa NXT RNGD Server"
- Rebellions(2024.08.), "Rebellions' Software Stack"
- Rebellions(2024.12.), "리벨리온-사피온 합병법인 공식 출범"
- Rebellions(2025.08.27.), "Rebellions Debuts REBEL-Quad at Hot Chips 2025, Breaking AI's Energy Tax with High-Performance Chiplet Innovation"

2. 국외문헌

- ABI Research(2025.05.28.), "NVIDIA's Strategy: Dominating AI Through Ecosystem, Access, and Interconnect"
- ACM SIGARCH(2018.04.), "John Hennessy and David Patterson Share ACM Turing Award"
- AIMultiple(2026.01.), "GPU Software for AI: CUDA vs. ROCm in 2026"
- AIMultiple(2026.01.), "LLM Inference Engines: vLLM vs LMDeploy vs SGLang"
- CoreWeave(2025.10.16.), "CoreWeave Unveils AI Object Storage, Redefining How AI Workloads Access and Scale Data"
- CSIS(2024.10.), "The Double-Edged Sword of Semiconductor Export Controls"
- European Commission(2024), "Case M.11766 - NVIDIA/Run:AI"
- Financial Times(2024.09), "Huawei's Bug-Ridden Software Hampers China's Efforts to Replace NVIDIA in AI"
- Financial Times(2024.12.), "Microsoft Acquires Twice as Many NVIDIA AI Chips as Tech Rivals"
- Financial Times(2025.05.), "Satellite Images Reveal Huawei's Advanced Chip Production Line in China"
- Gartner(2025.10.10.), "Gartner Says AI-Optimized IaaS Is Poised to Become the Next Growth Engine for AI Infrastructure"
- Gartner(2026.01.), "Gartner Says Worldwide AI Spending Will Total $2.5 Trillion in 2026"
- Google(2025.02.), "Build Production AI on Cloud TPUs with JAX"
- Google Cloud(2025), "Boosting LLM Performance With Tiered KV Cache on Google Kubernetes Engine"
- Google Cloud(2026), "Cloud TPU Inference"

- HPCwire(2025.09.), “The Fast and the FuriosaAI: Korean Chip Startup Takes Aim at NVIDIA GPUs”
- Hyperion Research, Intel, Ansys(2024.08.), “Understanding the Total Cost of Ownership in HPC and AI Systems”
- Intel(2022.06.01.), “Intel to Acquire Codeplay Software”
- Intel(2024.06.), “oneAPI: A Viable Alternative To CUDA Lock-in”
- Investing.com(2024.05.), “Morgan Stanley Upgrades SMIC on Rising Demand for China-Made AI Chips”
- Kwon, Woosuk, et al.(2023), “Efficient Memory Management for Large Language Model Serving with PagedAttention”
- Lee et al.(2024), “Debunking the CUDA Myth: Towards GPU-based AI Systems”
- Li et al.(2021), “The Deep Learning Compiler: A Comprehensive Survey”
- Linux Foundation(2023.11.19.), “Unified Acceleration Foundation Forms to Drive Open Accelerated Compute and Cross-Platform Performance”
- Liu et al.(2024), “Aceso: Efficient Parallel DNN Training through Iterative Bottleneck Alleviation”
- Liu, Yuhan et al.(2025), “LMCache: An Efficient KV Cache Layer for Enterprise-Scale LLM Inference”
- Mark Horowitz(2014.02.), “Computing’s Energy Problem (and What We Can Do About It)”
- MLCommons(2024.08.), “New MLPerf Inference v4.1 Benchmark Results Highlight Rapid Hardware and Software Innovations in Generative AI Systems”
- Merics(2025.07.), “China’s Drive Toward Self-Reliance in Artificial Intelligence”
- New York Times(2026.02.), “Racing to Catch Up With NVIDIA, AMD Signs Chips-for-Stock Deal With Meta”
- Norman P. Jouppi et al.(2017.06.), “In-Datacenter Performance Analysis of a Tensor Processing Unit”
- Norman P. Jouppi et al.(2023), “TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings”
- NVIDIA,(2011.05.10.), “Universities and Research Institutions Around the World Embrace Parallel Computing Using GPUs”
- NVIDIA(2025.01.), “NVIDIA AI Enterprise Software Stack”
- NVIDIA(2025.06.18.), “LLM Inference Benchmarking: How Much Does Your LLM Inference Cost?”
- NVIDIA(2026.02.19.), “LMCache Integration in Dynamo”
- NVIDIA, “Management Library (NVML)”, Documentation
- NVIDIA, “Multi-Instance GPU User Guide”, Documentation
- OECD(2025.11.), “Competition in Artificial Intelligence Infrastructure”
- Philippe Tillet et al.(2019), “Triton: An Intermediate Language and Compiler for Tiled Neural Network Computations”
- PyTorch(2024.09.), “vLLM Joins PyTorch Ecosystem: Easy, Fast, and Cheap LLM Serving for Everyone”
- PyTorch(2024.12.21.), “Rebellions Joins the PyTorch Foundation as a General Member”
- Reuters(2024.06.), “Huawei’s Harmony Aims to End China’s Reliance on Windows, Android”
- ROCm, “Unified Memory Management”, Documentation
- SGLang(2026), “SGLang Documentation”

- SIA(2025.12.), “Global Semiconductor Sales Q3 2025 & November 2025”
- Stanford HAI(2025.04.), “Artificial Intelligence Index Report 2025”
- Stanford University(2025.02.), “CS230 Deep Learning”
- Sze et al.(2024.03.), “Efficient Processing of Deep Neural Networks: A Tutorial and Survey”
- TechRadar(2025.08.), “Brave or Foolhardy? Huawei Takes the Fight to NVIDIA CUDA by Making Its Ascend AI GPU Software Open Source”
- TechNow(2025.11.), “Huawei CANN vs CUDA: Can Open-Source AI Break NVIDIA’s Grip?”
- Tianqi Chen et al.(2018.02.), “TVM: An Automated End-to-End Optimizing Compiler for Deep Learning”
- Tom’s Hardware(2025.02.), “DeepSeek Research Suggests Huawei’s Ascend 910C Delivers 60% of NVIDIA H100 Inference Performance”
- Tri Dao et al.(2022), “FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness”
- Visual Capitalist(2025), “Charted the Battle for AI Data Center Revenue 2021-2025”
- WSTS(2025.12.), “Autumn Forecast 2025-2026”