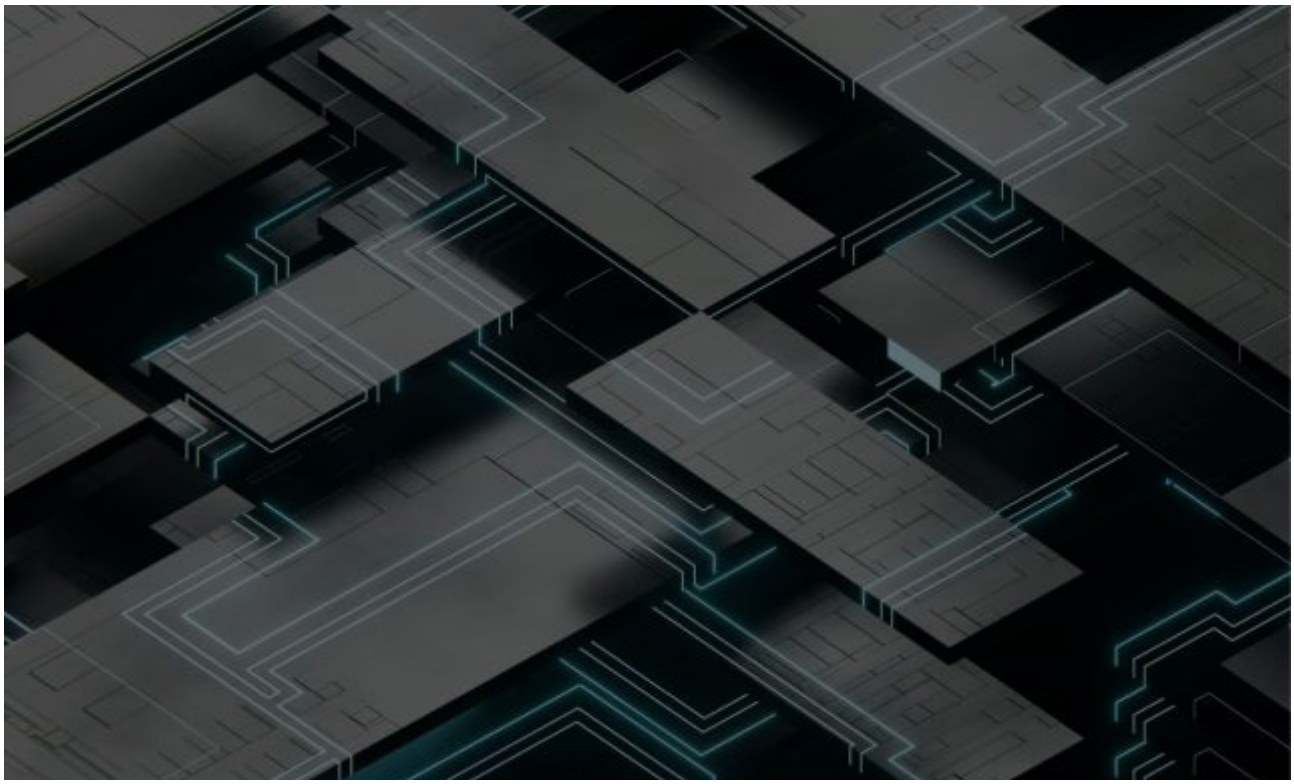
rebellions_

NPU 기반 LLM 서빙: 성능과 효율을 위한 새로운 시스템

Aug 21, 2025



들어가며

대규모 언어모델(LLM)의 시대에는 “모델을 실행하는 것”에서 “모델을 효율적이고 안정적으로, 그리고 대규모로 서빙하는 것”으로 초점이 옮겨지고 있습니다. 이제 성공의 핵심은 시스템 수준의 설계에 있습니다. 복잡한 추론 워크플로우 오버헤드를 최소화하면서 하드웨어에 최적화된 실행 경로로 재구성하는 것입니다.

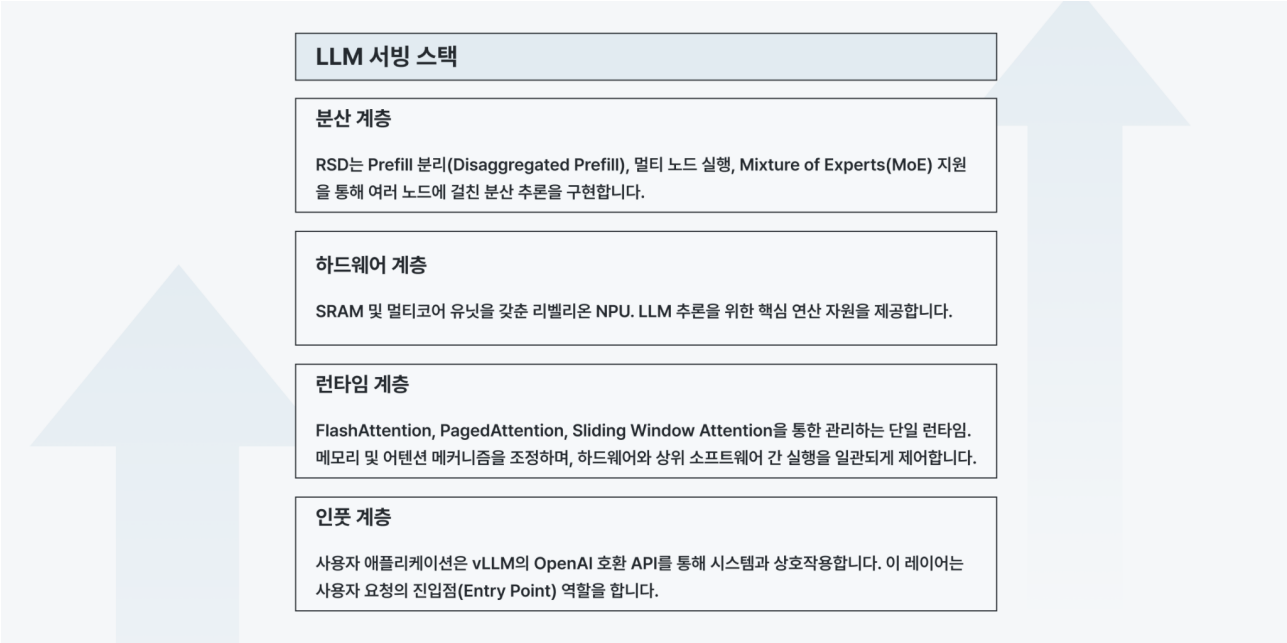
최신 LLM들은 FlashAttention, PagedAttention, Sliding Window Attention 등 고급 어텐션 메커니즘을 활용하여 높은 처리량, 낮은 지연, 긴 컨텍스트 지원을 달성합니다. vLLM과 같은 프레임워크들은 이 과정을 GPU 기반 서빙에 최적화시키면서 CUDA를 이용해 빠른 메모리 접근과 병렬 연산을 수행합니다. 하지만 GPU에 대한 높은 의존도는 전력 소모와 비용을 증가시키며 NPU 등 다른 아키텍처와의 호환성 또한 제한합니다.

리벨리온은 이러한 한계를 해결하기 위해, 위 메커니즘을 자체 NPU 아키텍처에 맞게 최적화했습니다. FlashAttention, PagedAttention, Sliding Window Attention을 리벨리온 NPU의 메모리 계층과 연산 구조에 맞춰 재설계하고, 이를 통합 런타임에 내장했습니다. vLLM RBLN 플러그인을 사용하면 이러한 최적화를 vLLM 워크플로우 내에서 그대로 활용하면서 기존 코드 수정 없이 NPU 기반의 성능과 효율을 그대로 누릴 수 있습니다.

리벨리온 NPU를 위한 LLM 서빙 최적화

리벨리온은 vLLM의 핵심 어텐션 메커니즘을 NPU 상에서 네이티브하게 실행되도록 재설계했습니다. FlashAttention과 PagedAttention은 하드웨어와 공동 설계되어 단일 런타임으로 통합되었으며, Causal Masking은 커널 단에서 처리되고, Scaled Dot-Product Attention (SDPA)은 최적화된 경로를 통해 지원됩니다. 이 구조는 모든 어텐션 타입에서 일관된 실행을 보장하며, 리벨리온 하드웨어에 특화된 성능을 제공합니다.

또한 리벨리온 NPU는 vLLM API와 완벽히 호환되어, 모델 배포 시 코드 변경 없이 하드웨어 수준의 메모리 접근·스케줄링·커널 실행 최적화를 활용할 수 있습니다.



[Figure 1. LLM 서빙 스택]

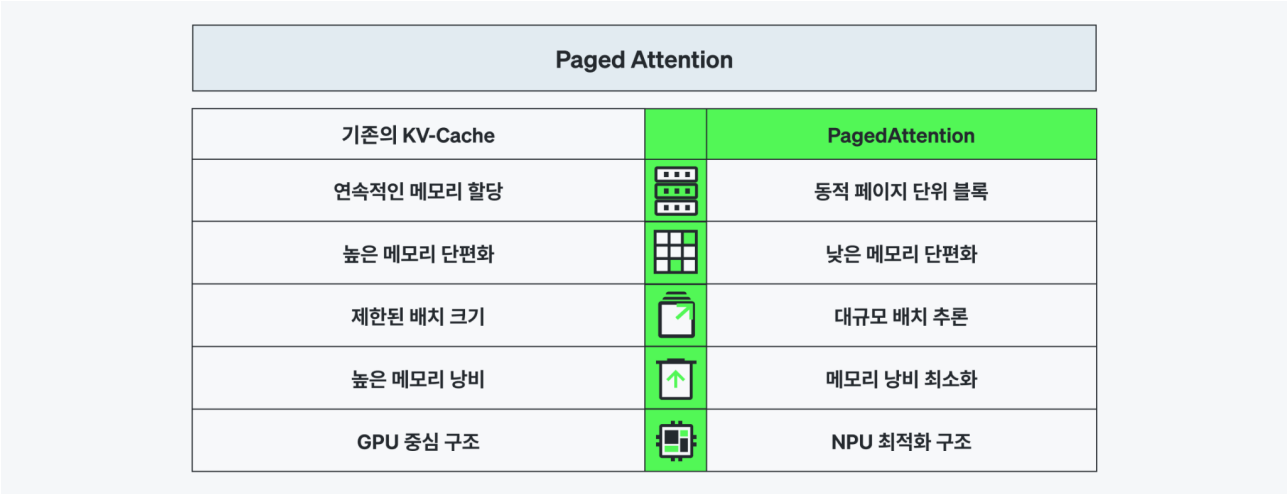
이 아키텍처는 리벨리온의 RSD (Rebellions Scalable Design)의 기반이 됩니다. RSD는 단일 디바이스를 넘어 멀티 노드·멀티 카드 환경으로 LLM 서빙을 확장하며, 프리필 분리(disaggregated prefill), Mixture of Experts (MoE) 라우팅 등을 지원해 확장성과 효율성을 동시에 확보합니다.

LLM 서빙을 위한 통합 실행 체계

FlashAttention

리벨리온의 FlashAttention은 NPU의 로컬 SRAM 크기에 맞춘 타일 기반 커널 구조로 구현되었습니다. Blockwise Softmax와 Matrix Multiplication은 공유 메모리(SHM) 내에서 전부 수행되어 DRAM 접근을 최소화하고 연산 효율을 극대화합니다.

실행 커널의 파티션 크기는 런타임에서 설정할 수 있으며, 사용자 지정 값 또는 optimum-rbln 라이브러리의 기본 설정을 통해 자동으로 선택됩니다. 이 라이브러리는 HuggingFace 모델을 Rebellions NPU와 연결하여 모델 컴파일 및 실행을 담당합니다. 정규화(Normalization)와 누적(Accumulation) 과정을 Fused Primitive 형태로 결합함으로써 DRAM–SHM 간 메모리 트래픽을 최소화했습니다.



[Figure 2. PagedAttention]

PagedAttention

PagedAttention은 KV 캐시를 논리 블록 단위로 관리해, 긴 시퀀스나 다중 세션 배치에서도 효율적인 메모리 사용을 가능하게 합니다. 기존의 Eager Attention과 달리, 디코딩 중 비활성 블록을 효율적으로 재활용·삭제하여 메모리 단편화(fragmentation)를 방지합니다.

리벨리온은 KV 블록 기반 커널 수준 메모리 관리로 이를 구현했습니다. 파티션 크기는 optimum-rbln을 통해 기본값으로 최적화되며, 성능과 메모리 사용량 간의 균형을 유지합니다.

런타임은 vLLM의 Block Table 구조와 완전히 호환됩니다. 추론 중 Block Table은 커널에 직접 전달되며, 커맨드 프로세서(Command Processor, CP)는 동적 DMA를 통해 주소를 실시간 평가하여 고정 주소에 의존하지 않고 임의의 DRAM 위치에 접근합니다. 이러한 동적 블록 어드레싱(Dynamic Block Addressing) 은 컴파일러의 런타임 주소 해석 기능을 기반으로 지원됩니다.

메모리 매핑, 블록 변환, 정렬은 모두 커널 내부에서 처리되어 비정형 워크로드에서도 사전 처리나 정적 할당 없이 **높은 추론 처리량**이 가능해집니다.

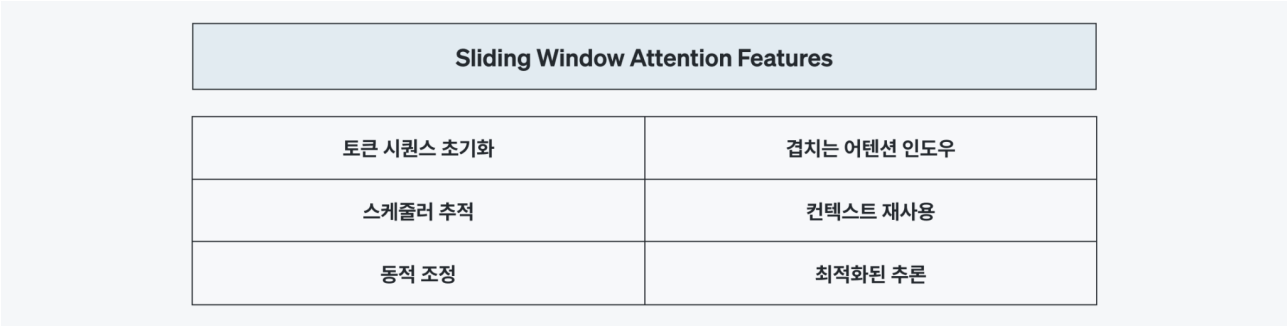
Causal Mask

Causal Masking은 연산 프리미티브(compute primitives) 내부에서 자동 처리됩니다. 따라서 런타임 중 별도의 어텐션 마스크를 생성하거나 전달할 필요가 없습니다. 이러한 단순화는 셋업 오버헤드를 줄이고, 오토리그레시브(autoregressive) 디코더 등 Causal Attention을 요구하는 모델에 대한 네이티브 지원을 가능하게 합니다.

Scaled Dot-Product Attention (SDPA)

리벨리온은 torch.nn.functional.scaled_dot_product_attention 인터페이스를 완전 지원합니다. 긴 시퀀스의 경우 메모리 및 연산 효율성을 위해 최적화된 파티션 크기를 사용하면서 FlashAttention을 **자동 적용**합니다. float, bool, None 타입의 마스크형 Attention 변형도 모두 내부적으로 NPU에 최적화되어 실행됩니다.

Sliding Window Attention

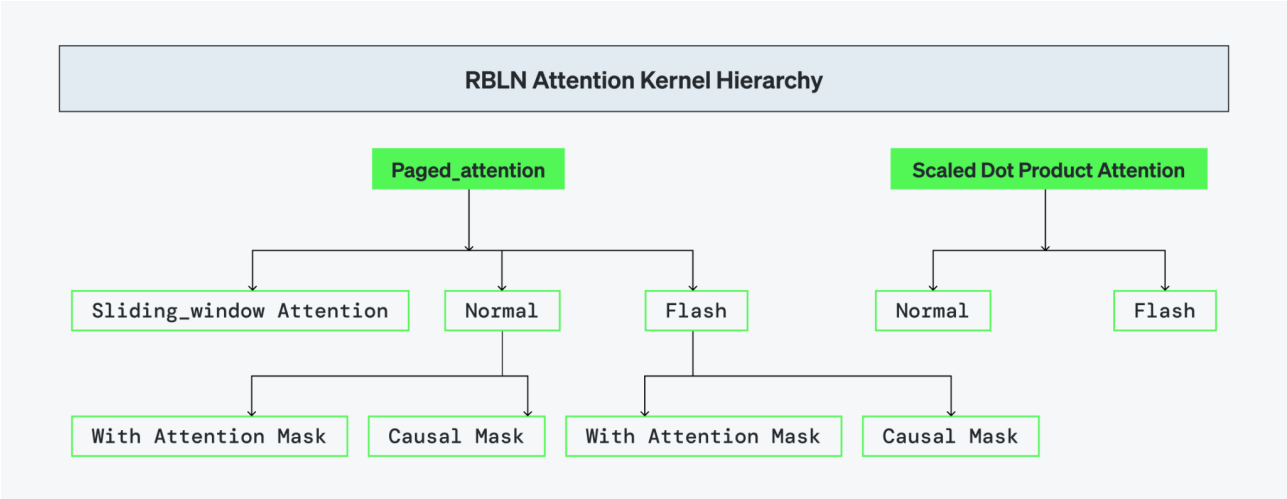


[Figure 3. Sliding Window Attention]

Sliding Window Attention은 고정 크기의 윈도우만 유지하면서 스트리밍 및 장문 맥락 추론을 가능하게 합니다. 전체 히스토리를 저장하는 대신, 현재 단계에 필요한 최신 토큰만 유지하여 메모리 사용량을 크게 절감합니다.

이것이 가능한 것은 효율적인 KV 캐시 윈도우 관리 방식 덕분입니다. 활성 윈도우만 메모리에 유지하고, 컨텍스트가 진행될 때 메모리 재할당 없이 기존 KV 엔트리를 제자리(in-place) 회전시킵니다.

런타임은 인덱스 회전과 윈도우 추적을 자동으로 관리하여 Gemma3 같은 모델도 일관된 메모리 사용량과 높은 토큰 처리량으로 동작할 수 있습니다. 이를 통해 리벨리온 NPU는 스트리밍 워크로드에서도 안정적으로 실행됩니다.



[Figure 4. RBLN Attention Kernel 체계]

vLLM RBLN Plugin

vLLM RBLN 플러그인은 어텐션 메커니즘들을 단일 실행 경로로 통합한 인터페이스로, 사용자 애플리케이션과 NPU 런타임을 연결합니다. FlashAttention과 PagedAttention은 공통 연산 그래프 및 메모리 모델을 공유하며 런타임에 내장되어 있습니다. 따라서 vLLM 기반 애플리케이션은 코드 변경 없이 NPU 특화 최적화(고처리량·저지연) 를 바로 활용할 수 있습니다.

현재 vLLM-RBLN 플러그인은 optimum-rbln과 통합되어 있습니다. 모델은 optimum-rbln으로 컴파일된 후 vLLM에서 모델 파라미터로 참조되는 구조입니다. 모든 온라인 튜토리얼이 이 기본 구현 방식을 기반으로 하고 있습니다.

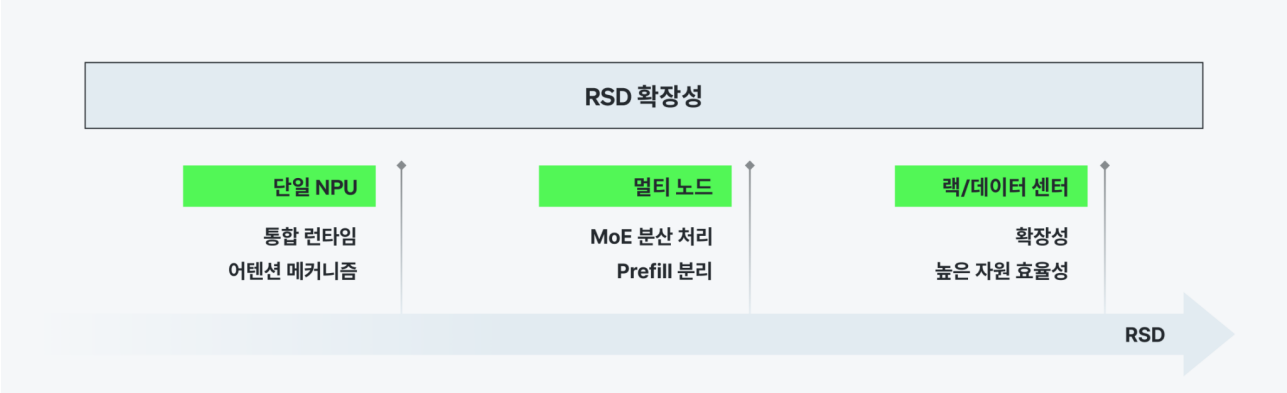
리벨리온은 현재 vLLM의 기존 API 및 모델 저장소(Model Zoo) 와 네이티브하게 통합되는 torch.compile() 기반의 차세대 아키텍처를 적극적으로 개발 중입니다. 이 설계는 별도의 컴파일 단계를 제거하여 표준 vLLM 워크플로우를 통해 보다 매끄러운 사용자 경험을 제공합니다. torch.compile()을 사용할 경우, 첫 실행 시 Cold Start 단계에서 모델이 자동 컴파일되고, 이후 실행부터는 Warm Start로 전환되어 캐시된 최적화 아티팩트(Optimized Artifacts) 를 활용하게 됩니다.

통합된 실행 중심 아키텍처

이 모든 구성 요소는 단순한 기능의 집합이 아니라, 완전히 통합된 실행 시스템을 이룹니다. 리벨리온의 런타임은 동적 시퀀스 배칭, 토큰 단위 병렬 처리, 캐시 인식 스케줄링, 메모리 컴팩션 등을 지원하며, 실제 서비스 환경에 맞춘 엔드투엔드 LLM 서빙 인프라로 동작합니다.

모든 연산은 단일 실행 그래프 내에서 관리되며, 연산 커널·메모리 전송·캐시 상태가 하나의 제어 경로로 통합되어 있습니다. 이러한 일체형 설계는 NPU 아키텍처 단계에서부터 최적화되어, 어텐션 메커니즘 간의 상호작용을 원활하게 하고 레이어드 확장 방식에서 발생하는 비효율을 근본적으로 제거합니다. 그 결과, 리벨리온 런타임은 실제 프로덕션 환경에 최적화된 LLM 서빙 성능을 구현하며, 시스템 전반의 일관성과 효율성을 극대화합니다.

RSD를 통한 확장성



[Figure 5. RSD 확장성]

LLM 서빙의 상용화를 위해서는 분산 아키텍처가 필수적입니다. RSD(Rebellions Scalable Design)는 확장된 LLM 서빙을 지원하는 기술 프레임워크로, 다음 기능을 제공합니다:

- **프리필 분리:** 컨텍스트 빌드(Prefill)와 디코딩(Decoding)을 분리하여 노드 간 자원 사용을 최적화
- **멀티 노드 실행:** 여러 NPU에 걸친 분산 추론을 통해 선형 확장성과 처리량 향상 구현
- **Mixture of Experts (MoE):** 전문가(Expert) 연산을 장치 간 효율적으로 분산 처리

이 구조를 기반으로 LLM 인스턴스는 여러 장치로 확장되고, 메모리 제약 하에서도 처리량을 유지하고, 자원별 부하를 지능적으로 분배합니다.

결론: 리벨리온의 AI 서빙 인프라

LLM 추론의 본질은 단순한 속도가 아니라 실행에 있습니다. 복잡한 모델과 다양한 워크로드를 안정적으로, 확장 가능하게 운영하려면 견고한 서빙 인프라가 필요합니다. 리벨리온은 이를 자사 NPU 기반에서 완전히 재설계된 풀스택 구조로 구현했습니다. FlashAttention, PagedAttention, Sliding Window Attention을 vLLM RBLN 플러그인을 통해 통합하여, vLLM 애플리케이션이 그대로 NPU 성능을 활용할 수 있습니다.

RSD는 이를 분산 환경으로 확장하여, Prefill 분리·멀티 노드 실행·MoE 지원을 통해 단순 하드웨어가 아닌 AI 서빙 인프라를 제공합니다. AI의 미래는 단순한 모델을 넘어선 확장 가능하고 실행 가능한 서빙 인프라 구축에 달려 있습니다. 리벨리온은 그 인프라를 구현하고 있습니다.