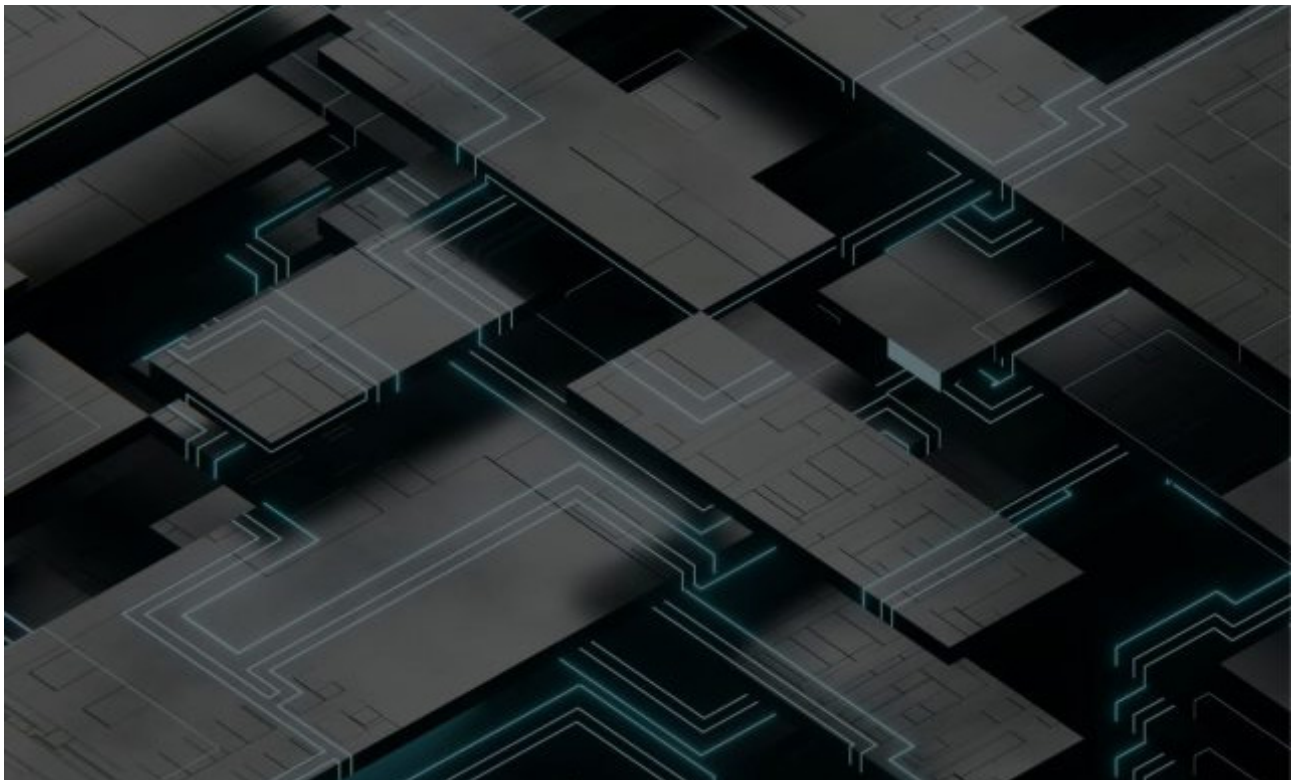
v.08 White Papers

rebellions_

NPU基盤のLLMサービング：性能と効率向けの新しいシステム

8月 21, 2025



序論

大規模言語モデル（LLM）時代には「モデルを実行すること」から「モデルを効率的かつ安定的に、さらに大規模でサービングすること」へと焦点が移りつつあります。もはや、成功のカギはシステムレベルの設計にかかっています。複雑な推論ワークフローのオーバーヘッドを最小化すると同時に、ハードウェアに最適化した実行パスに再構成する必要があります。

最新のLLMはFlashAttention、PagedAttention、Sliding Window Attentionなど高レベルのアテンションメカニズムを活用して、高いスループット（Throughput）、低いレイテンシ（Latency）、長いコンテキスト（Long Context）を実現しています。vLLMのようなフレームワークは、こうした過程をGPU基盤のサービングに最適化しながら、CUDAを利用して高速なメモリアクセスと並列演算を行います。しかし、GPUへの高い依存度は電力消費と費用を増加させ、NPUなど他のアーキテクチャとの互換性にも制限がかかります。

リベリオンはこのような限界を乗り越えるために、上記のメカニズムを独自のNPUアーキテクチャに合わせて最適化しました。FlashAttention、PagedAttention、Sliding Window AttentionをリベリオンNPUのメモリ階層と演算構造に合わせて設計し、これを統合ランタイムに搭載しました。 vLLM RBLNのプラグインを使用すると、この最適化をvLLMのワークロード内でそのまま活用できるうえ、既存のコードを修正しなくてもNPU基盤の性能と効率を同じく維持できます。

リベリオンNPU向けのLLMサービングの最適化

リベリオンは、vLLMの中核となるアテンションメカニズムをNPU上でネイティブに実行できるように見直しました。FlashAttentionとPagedAttentionは、ハードウェアと共同設計し、単一ランタイムで統合しました。また、Causal Maskingはカーネルレベルで処理され、Scaled Dot-Product Attention（SDPA）は最適化したパスを通じてサポートします。この構造は全てアテンションタイプで一貫した実行ができるようにし、リベリオンハードウェアに特化した性能の提供が可能です。

また、リベリオンNPUはvLLM APIと完全互換し、モデルをデプロイする際にもコードを変更せずに、ハードウェアレベルのメモリアクセス・スケジューリング・カーネル実行の最適化を利用できます。

[Figure 1．LLMのサービングスタック]

このアーキテクチャはリベリオンのRSD（Rebellions Scalable Design）の基盤となります。RSDは単一デバイスを超え、マルチノード・マルチカード環境でLLMサービングを拡張し、Prefill分離（disaggregated prefill）、Mixture of Experts（MoE）ルーティングなどに対応し、拡張性と効率性を同時に確保しています。

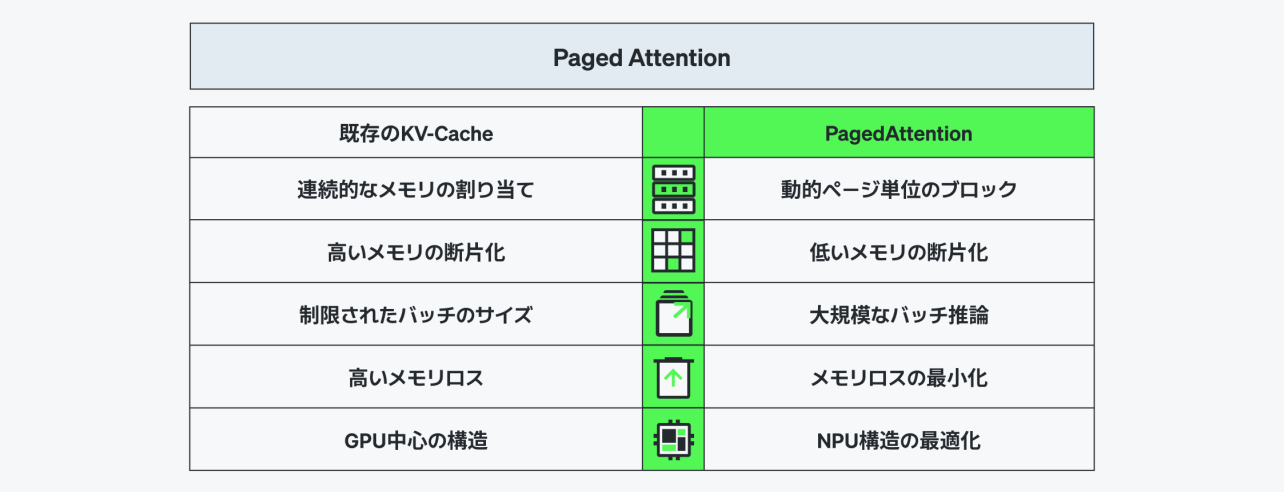
LLMサービング向けの統合実行体系（Unified Execution for LLM Serving)

FlashAttention

リベリオンのFlashAttentionは、NPUのローカルSRAMのサイズに合わせたタイルベースのカーネル構造で実現されています。Blockwise SoftmaxとMatrix Multiplicationは、共有メモリ（SHM）内で全て動作されます。また、DRAMへのアクセスを最小化し演算効率を最大化します。

実行カーネルのパーティションのサイズはランタイムで設定でき、ユーザーの指定値やoptimum-rblnライブラリーの基本設定で自動的に選択されます。このライブラリーはHuggingFaceモデルをRebellions NPUとつなげ、モデルコンパイルおよび実行を担います。正規化（Normalization）と累積（Accumulation）の過程をFused Primitiveの形式で

結合することで、DRAM–SHM間のメモリトラフィックを最小限に抑えました。



[Figure 2. PagedAttention]

PagedAttention

PagedAttentionはKVキャッシュを論理ブロック単位で管理し、長いシークエンスや多層セッションのバッチでもメモリを効率よく使えます。既存のEager Attentionとは違い、デコード中の非活性ブロックを効率的に再利用・削除しメモリの断片化（fragmentation）を防ぎます。

リベリオンはKVブロック基盤のカーネルレベルのメモリ管理でこれを実現しました。パーティションのサイズはoptimum-rblnによりデフォルトを最適化し、性能とメモリ使用量間のバランスを維持しています。

ランタイムはvLLMのBlock Table構造と完全互換します。推論中のBlock Tableは、カーネルに直接送信し、Command Processor（CP）は動的DMAを通じてリアルタイムでアドレスを評価し、固定したアドレスに依存せずに、任意のDRAMにアクセスします。こうした動的ブロックアドレッシング（Dynamic Block Addressing）は、コンパイラのランタイムアドレスの解析機能を基にサポートされます。

メモリマッピング、ブロック変換、アラインメントは、すべてカーネル内部で処理され、非定型ワークロードでも事前処理や静的な割り当てなしでも推論スループット（Throughput）が高くなります。

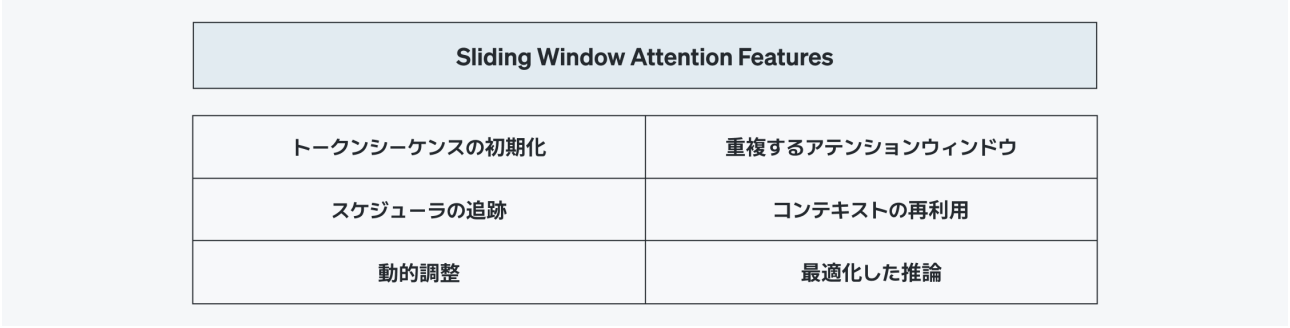
Causal Mask

Causal Maskingは演算プリミティブ（compute primitives）内部で自動処理されます。よって、ランタイム中に別途のアテンションマスクを生成したり、伝達する必要がありません。このような単純化はセットアップオーバーヘッドを減らし、オートレグレッシブ（autoregressive）デコーダーなどCausal Attentionが求められるモデルに対してもネイティブ対応ができます。

Scaled Dot-Product Attention (SDPA)

リベリオンはtorch.nn.functional.scaled_dot_product_attentionのインターフェースに完全対応します。長いシークエンスの場合、メモリおよび演算の効率性のために最適化したパーティションのサイズを使いながら、FlashAttentionを自動的に適用します。Float、bool、Noneタイプのマスク型Attentionの変形も全て内部でNPUに最適化して実行されます。

Sliding Window Attention Features

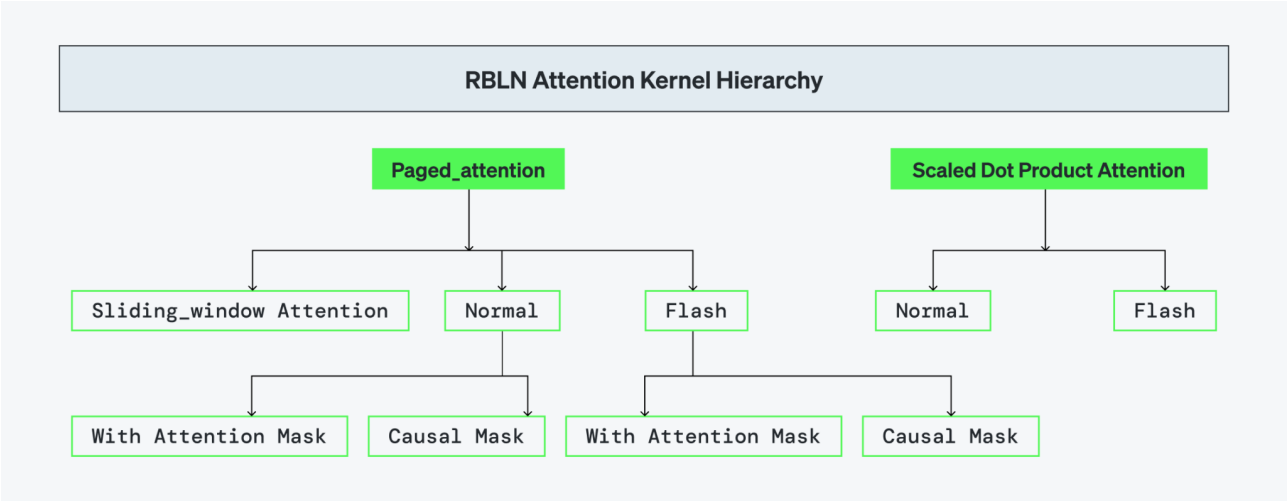


[Figure 3. Sliding Window Attention]

Sliding Window Attentionは固定サイズのウィンドウのみ維持しながら、ストリーミングおよび長文コンテキストでの推論が可能です。全体のヒストリーを保存する代わりに、現段階で必要な最新のトークンだけを維持するため、メモリの使用量を大幅に減らすことができます。

これが可能なのは、KVキャッシュウィンドウを効率的に管理しているからです。メモリには活性ウィンドウだけを維持し、コンテキストが進行する間もメモリを再割り当てすることなく、既存のKVエントリーをその場（in-place）で回転させます。

ランタイムはインデックス回転とウィンドウ追跡を自動管理し、Gemma3のようなモデルも一貫したメモリ使用量と高いトークン処理量で動作できます。これにより、リベリオンNPUはストリーミングワークロードでも安定的に実行できます。



[Figure 4. RBLN Attention Kernalの構造]

vLLM RBLN Plugin

vLLM RBLNプラグインは、アテンションメカニズムを単一の実行パスで統合したインターフェースであり、ユーザーアプリケーションとNPUランタイムをつなげます。FlashAttentionとPagedAttentionは、共通の計算グラフおよびメモリモデルを共有し、ランタイムに搭載されています。そのため、vLLM基盤のアプリケーションはコードを変更せずにNPUに特化した最適化（高処理量・低遅延）を直ちに利用できます。

現在、vLLM-RBLNプラグインはoptimum-rblnと統合しています。モデルはoptimum-rblnでコンパイルした後、vLLMでモデルパラメータによって参照される仕組みです。すべてのオンラインチュートリアルはこの基本的な実現方式に基づいています。

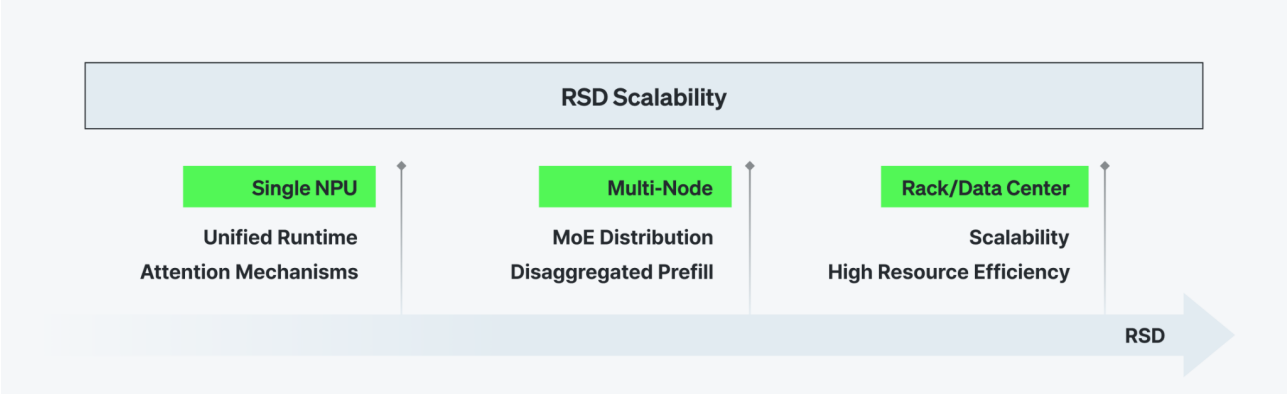
リベリオンは現在、vLLMの既存APIおよびモデルリポジトリ（Model Zoo）とネイティブに統合されるtorch.compile()基盤の次世代アーキテクチャを積極的に開発しています。この設計は別途のコンパイル段階を省き、標準vLLMのワークフローを通じてより円滑なユーザー経験が可能です。torch.compile()を使う場合は、最初に実行する際に、Cold Start段階でモデルが自動的にコンパイルされます。この後の実行からは、Warm Startに切り替わってキャッシュした最適化アーティファクト（Optimized Artifacts）を用います。

統合型実行アーキテクチャ

全てのコンポーネントは単なる機能の集合ではなく、完全に統合された実行システムを構成しています。リベリオンのランタイムは動的なシーケンスバッチ、トークン単位の並列処理、キャッシュ認識のスケジューリング、メモリコンパクションなどに対応し、実際のサービス環境に合わせたエンドツーエンドのLLMサービングインフラで動作できます。

全ての演算は単一の実行グラフ内で管理され、演算カーネル・メモリ転送・キャッシュの状態が一つの制御パスで統合されています。この一体型設計は、NPUアーキテクチャの段階から最適化されており、アテンションメカニズム間の相互作用を円滑にします。さらに、レイヤードの拡張で発生する非効率を根本的に除去します。その結果、リベリオンのランタイムは実際のプロダクション環境で最適なLLMサービング性能を実現し、システム全体の一貫性と効率性の最大化が可能です。

RSDによる拡張性



[Figure 5. RSDの拡張性]

PrLLMサービングの商用化には分散アーキテクチャが欠かせません。RSD（Rebellions Scalable Design）は拡張したLLMサービングに対応する技術フレームワークで、以下の機能を提供しています：

- Disaggregated Prefill：コンテキストビルド（Prefill）とデコード（Decoding）を分離し、ノード間の稼働率を最適化
- マルチノードの実行：複数のNPUにわたる分散推論によりスケーラビリティとスループットの向上を実現
- Mixture of Experts（MoE）：専門家（Expert）の演算をデバイス間で効率的に分散処理

この構造を基にLLMインスタンスは複数のデバイスに拡張され、メモリの制限があってもスループットを維持し、リソース別の負荷を賢く配分します。

結論：リベリオンのAIサービングインフラ

LLM推論の本質は、単なる速度ではなく実行にあります。複雑なモデルと多様なワークロードを安定的に拡張しながら運営するには、 堅牢なサービングインフラが必要です。リベリオンはこのインフラを自社のNPU基盤で完全に再設計されたフルスタック構造で実現しました。 FlashAttention、PagedAttention、Sliding Window AttentionをvLLM RBLNプラグインで統合し、vLLMアプリケーションでそのままNPUの性能を利用できます。

RSDはこれを分散環境に拡張し、Prefillの分離・マルチノードの実行・MoEにも対応しています。これにより、単なるハードウェアではなくAIサービングインフラを提供しています。AIの未来はモデルに止まらず、拡張と実行が可能なサービングインフラの構築にかかっています。リベリオンはまさにこのインフラを構築しています。