vAttention: Dynamic Memory Management for Serving LLMs without PagedAttention

Akira Kawata

1 Introduction

1 Introduction - Paged Attention の登場

- LLM の推論が重要
- LLM の推論においてはメモリ上の kv cache と呼ばれる領域がコンテキスト長に比例して伸びる
  - TODO: ml-meccha-wakaru へのリンク
- 考えられる最大長の kv cache を事前に割り当てるシステムもあるが、これではメモリが無駄遣いされてしまう
- そこで kv cache をブロックに分割したうえで、オンデマンドに kv cache ブロックを割り付ける PagedAttention が登場し、デファクトスタンダードになった

Some figures from PagedAttention paper

- OS の仮想メモリ機構をユーザ空間で実装している

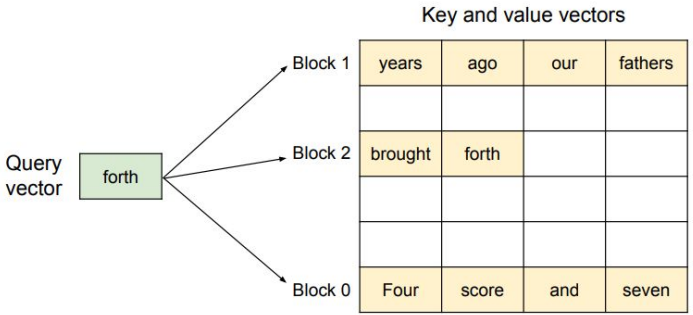

Figure 5. Illustration of the PagedAttention algorithm, where the attention key and values vectors are stored as non-contiguous blocks in the memory.

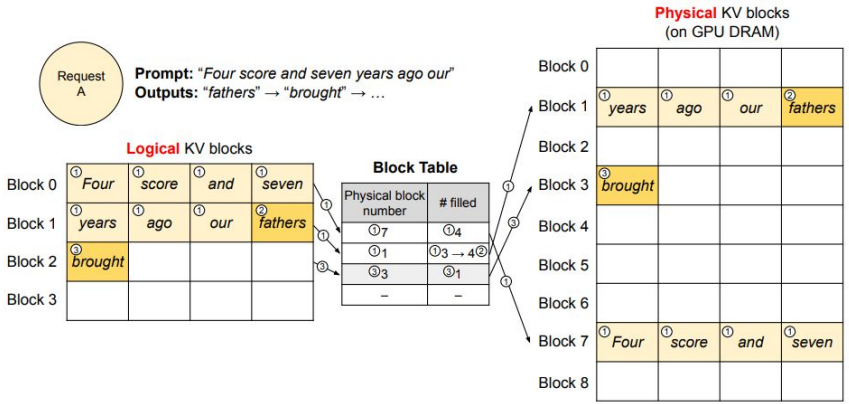

Figure 6. Block table translation in vLLM.

1 Introduction - PagedAttentionの問題点

- Attentionカーネルの書き換えが必要になる
  - kv cache が非連続になっているので修正が必要になる
- 独自のメモリマネージャーの実装が必須になる
- CPUおよびGPU実行時のランタイムオーバーヘッドがある

vAttention

- vAttention とは何?
  - vAttentionは、物理メモリの断片化を軽減しつつ、KVキャッシュの仮想メモリ連続性を保持するアプローチです
  - これは、CUDA仮想メモリ管理 (VMM) APIを利用して、仮想メモリと物理メモリの割り当てを分離することで実現されます。従来のcudaMallocとは異なり、これにより物理メモリは必要になったときにのみ割り当てられます
- メリット
  - 様々なAttentionカーネルをそのままサポートできる

vAttention を実現するうえでの問題点

- メモリ割り当てのレイテンシが大きい
  - メモリ割り当てと計算のオーバーラップ
  - 投機的先行割り当て
  - メモリ解放の遅延
- ページ粒度が大きい
  - CUDAはデフォルトで2MB単位でのメモリ割り当てしかサポートしておらず、これが大きな断片化を引き起こす可能性があります
  - vAttentionは、オープンソースのCUDA統合仮想メモリドライバーを修正し、より小さい64KBページのサポートを追加することでこれに対処します

2 Background

2.1 Large Language Models

- Skip

2.2 Fragmentation and PagedAttention

- LLM推論におけるメモリ割り当ての課題
  - 大規模言語モデル (LLM) の推論において、KVキャッシュ（処理されたトークンのアクティベーションを保存するGPUメモリ領域）の効率的な割り当ては非常に困難です
- PagedAttentionによる解決策
  - PagedAttentionは、KVキャッシュを固定サイズのブロックに分割し、必要に応じてGPUメモリの小さなブロックを動的に割り当てることで、メモリ断片化の問題をほぼ完全に解決しました

3 Issues with the PagedAttention Approach

3.1 Requires Re-writing the Attention Kernel

- 従来の Attention 機構は、入力テンソルKとVが連続したメモリに保存されていることを前提としている
- PagedAttention を利用するためには、非連続なKVキャッシュブロックを利用できるように書き換える必要がある
- https://github.com/vllm-project/vllm/blob/7b1895e6ce4942091e16da790af8c12772a1d384/csrc/attention/attention_kernels.cuh#L227

3.2 Adds Redundancy in the Serving Framework

- PagedAttentionでは、vLLM がKVキャッシュと動的に割り当てられたメモリブロック間のマッピングを管理する
- vLLM は、これらのKVキャッシュブロックの仮想メモリアドレスを追跡し、実行時にアテンションカーネルに渡す必要がある
- つまり、オペレーティングシステムが仮想-物理アドレス変換のためにすでに行っていることを実質的に重複して実装することを要求する

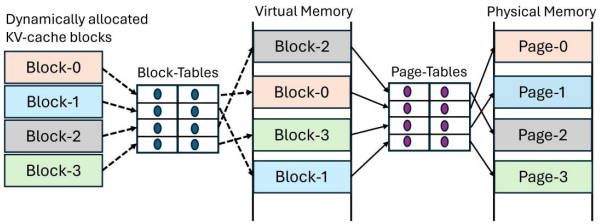

Figure 1. PagedAttention involves two layers of memory management: one in user space and one in OS kernel space.

3.3.1 Runtime overhead on the GPU

- vLLMでは、Block-Tableのルックアップと追加の分岐のオーバーヘッドにより、PagedAttentionベースの実装が対応する非PagingなFasterTransformerカーネルよりも20〜26%遅いことが確認されている

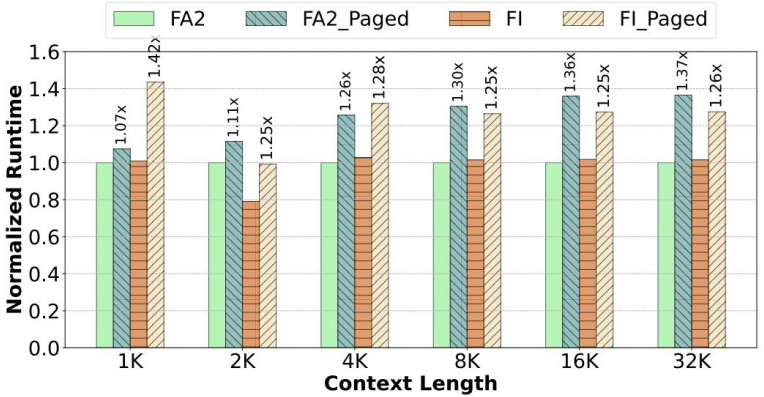

Figure 2. Overhead of PagedAttention in prefill kernels (model: Llama-3-8B, one A100 GPU). Numbers on top show overhead over the corresponding non-paged implementation of FlashAttention-2 (FA2) and FlashInfer (FI).

3.3.2 Runtime overhead on the CPU

- 追加のメモリマネージャーを実装すると、サービングシステムのCPU ランタイムでパフォーマンスの問題が発生する可能性がある
- vLLMでは、Block-Tableの準備に要するレイテンシがバッチ構成に依存し、max_num_blocks × batch_sizeに比例して増加する

4 Insights into LLM Serving Systems

4 Insights into LLM Serving Systems

- KVキャッシュのメモリ要件は反復ごとに予測可能である
  - オートリグレッシブなデコードフェーズに入ると、LLMリクエストのKVキャッシュサイズは反復ごとに一律に1トークンずつ増加する
- KVキャッシュは高いメモリ割り当て帯域幅を必要としない
  - シングルGPU上でYi-6B、および2つのA100 GPU上でLlama-3-8BとYi-34Bを動作させた実験から、トークンあたりのメモリフットプリントは一般的に数十〜数百キロバイトであることが示されている
  - 各反復は通常、数十〜数百ミリ秒かかるため、リクエストが1秒あたりに必要とするメモリは最大で数メガバイトである

5 vAttention: Design and Implementation

5.1 Design Overview

- vAttentionのポイントは、物理メモリの断片化を、KVキャッシュの仮想メモリにおける連続性を維持したまま回避できるという点である。
- これを実現するために、vAttentionはCUDA仮想メモリ管理（VMM）APIを活用して、仮想メモリの割り当てを物理メモリの割り当てから分離することでデマンドページングのシステムサポートを活用している。
- この設計では、KVキャッシュ用の大きな連続したバッファを仮想メモリに事前に割り当て、物理メモリの割り当ては実行時まで延期する。
- 仮想メモリは現代の64ビットシステムにおいて潤沢であるため、仮想メモリの断片化や浪費は問題とならない

5.2 Leveraging CUDA Virtual Memory Support

- 標準のGPUメモリ割り当てインターフェースである cudaMalloc は、仮想メモリと物理メモリを同時に割り当てるため、デマンドページングをサポートしない。
- しかし、近年のCUDAバージョンでは、プログラマに対して仮想メモリと物理メモリの管理に関するよりきめ細かい制御が提供されており、それらの割り当ての分離もサポートされている。
- vAttentionは、これらの低レベルAPIを活用している

5.2.1 CUDA virtual memory APIs. (1)

- 表3 に、仮想メモリと物理メモリの割り当てを分離可能にするCUDA VMM APIの概要が示されている。
- 割り当ての粒度はGPUが使用するページサイズに依存する。
- 仮想メモリバッファまたは物理メモリハンドルのサイズは、物理メモリ割り当て粒度の倍数でなければならない。
- 物理メモリページは、仮想メモリバッファ内のサブ領域に対して、他のサブ領域とは独立して割り当て（または解放）できる。
- CUDA API（cuで始まる）は2MBページのみをサポートするが、vAttentionのCUDA拡張API（vで始まる）は、よりきめ細かい割り当てをサポートする

5.2.1 CUDA virtual memory APIs. (2)

CUDA VM APIs	vAttention VM APIs	Description	Latency (microseconds) 64KB	128KB	256KB	2MB
cuMemAddressReserve *	vMemReserve *	Allocate a buffer in virtual memory	18	17	16	2
cuMemCreate *	vMemCreate *	Allocate a handle in physical memory	1.7	2	2.1	29
cuMemMap	vMemMap	Map a physical handle to a virtual buffer	8	8.5	9	2
cuMemSetAccess	-	Enable access to a virtual buffer	-	-	-	38
cuMemUnmap	-	Unmap physical handle from a virtual buffer	-	-	-	34
cuMemRelease *	vMemRelease *	Free physical pages of a handle	2	3	4	23
cuMemAddressFree *	vMemFree *	Free a virtual memory buffer	35	35	35	1

Table 3. CUDA VMM APIs. * represents APIs that we use once while instantiating or terminating the serving framework. Rest of the APIs are used for (un)mapping physical memory pages at runtime. CUDA APIs (prefixed with cu) support only 2MB pages, whereas our CUDA extension APIs (prefixed with v) support fine-grained allocations.

CUDA Driver API - 6.14. Virtual Memory Management

Skip 5.2.2 to 5.3.4

6 Optimizations

6.1 Hiding Latency of Memory Allocation

- 演算とのメモリ割り当てのオーバーラップ（デコードフェーズ）
  - デコードフェーズでは、各イテレーションで1つの出力トークンのみが生成されるため、メモリ要求が事前に予測可能である
  - vAttentionは、リクエストがいつより多くのメモリを必要とするかを判断し、先行するイテレーションが実行されている間にバックグラウンドスレッドを使用して新しいページグループを割り当てる
- 遅延解放 + 事前割り当て（プリフィルフェーズ）
  - リクエストR1が完了し、新しいリクエストR2がバッチに参加する場合、vAttentionはR1のページグループの解放を遅延させR2に割り当てる

6.2 Mitigating Internal Fragmentation

- 課題: cuMemCreateなどのCUDA VMM APIは、現在2MBの大きなページ単位でのみメモリを割り当てるため、内部断片化を引き起こす可能性がある。
- 解決策: vAttentionは、既存のCUDA VMM APIと同じ機能を提供しつつ、複数のページサイズをサポートする新しいAPIセットをNVIDIAのオープンソースドライバーに実装する

Skip 7, 8, 9, 10 sections