Keep the Cost Down: A Review on Methods to Optimize LLM's KV Cache Consumption

Shi Luohe & Zhang Hongyi — Wuhan University
Yao Yao — Shanghai Jiao Tong University
Li Zuchao — Wuhan University
Zhao Hai — Shanghai Jiao Tong University

COLM 2024

2024/11/12

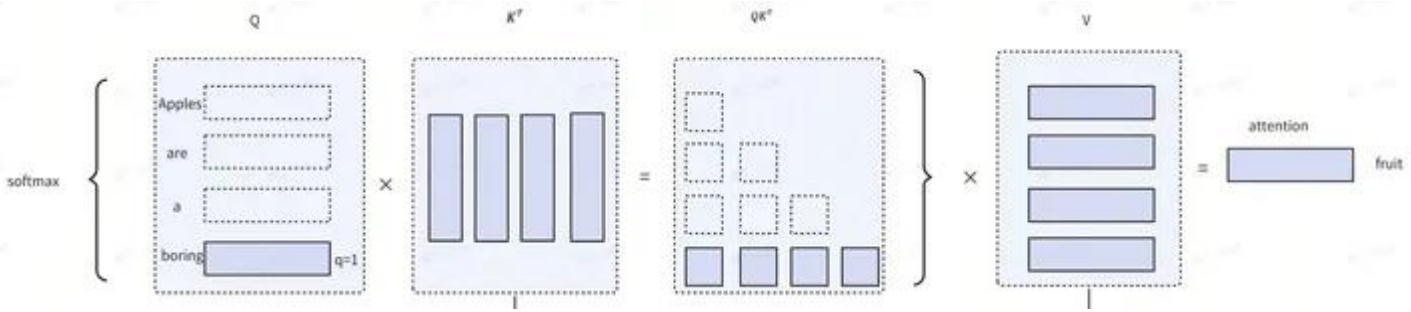
汇报人：吴亚辰

Contents

Part 01

What is KV-Cache And Why Need Opt

What is KV Cache?



Key-Value Paires Cache
键值对缓存

Reduce Redundant Computation
减少冗余计算

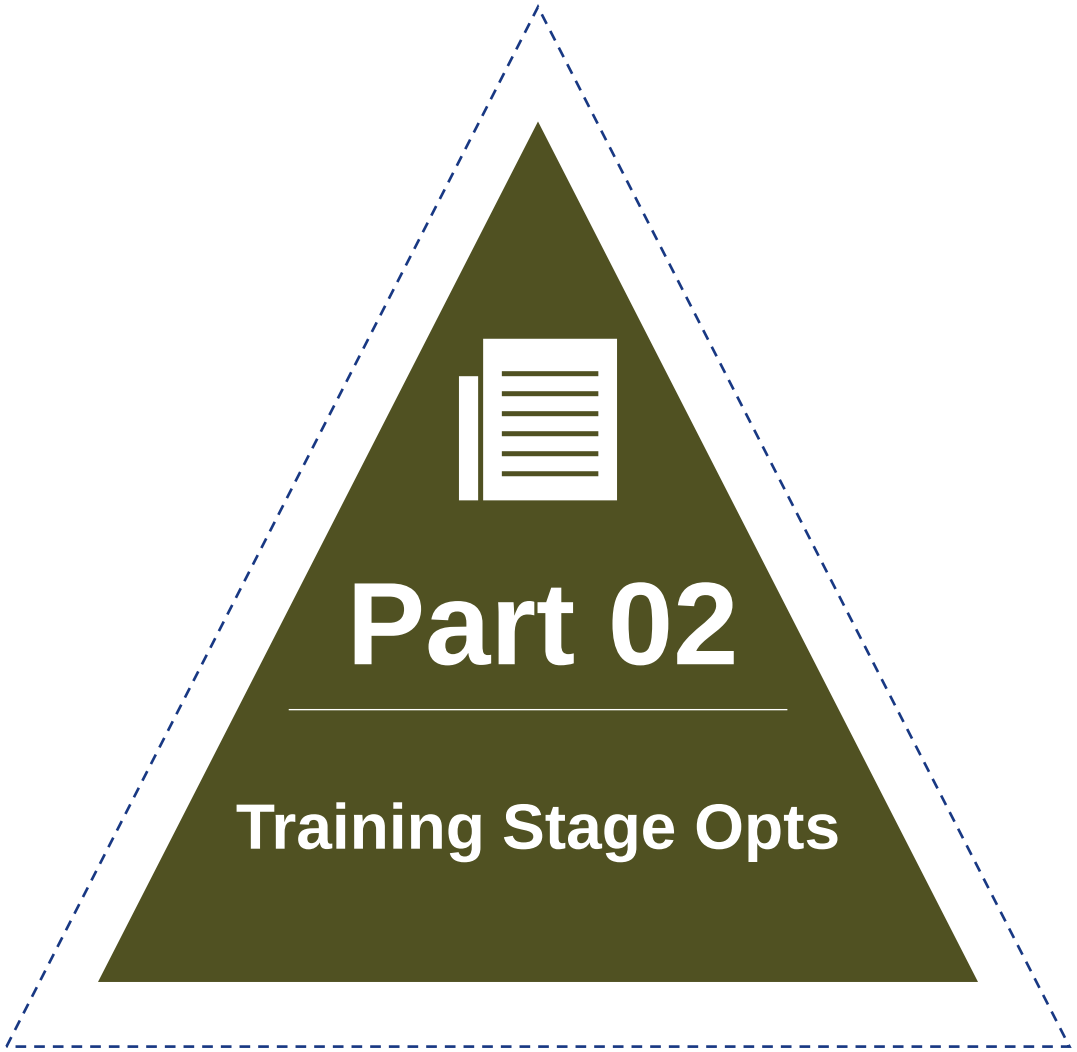
Inference Optimization
优化推理过程

KV Cache 是一种用于存储和重用先前计算的键值对（Key-Value Pairs）的技术，主要应用于注意力机制中，通过减少重复计算和优化内存管理，提高模型的推理和训练效率。

Why Should We optimize it?

随着序列长度增加，KV Cache的容量也会线性增长!

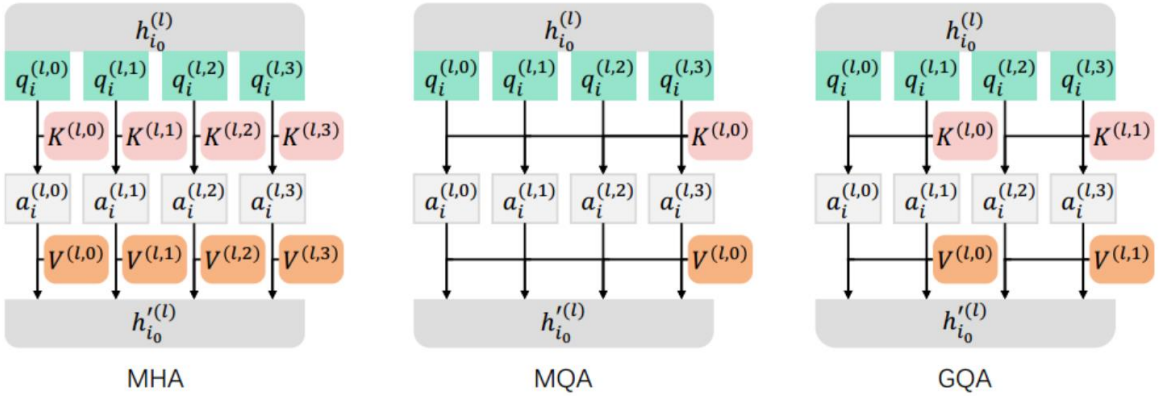
所谓优化KV Cache，就是研究如何reduce memory space，在不弱化或者不显著弱化模型的能力的前提下，降低或者优化KV Cache占用的存储空间。



Part 02

Training Stage Opts

MHA. MQA. GQA



MQA:

*对于所有的Keys-Values只保留**一个头**，实际上就是Wk和Wv权重矩阵把Embedding进行降维以匹配每一个Query头的维度，可以看作是一种 **low-rank低秩** 表示。*

KV Cache的空间占用率是原先MHA的 1/nh

GQA:

*对于所有的Keys-Values保留**多个(ng 个)头***
而且1<=ng<=nh，当ng=1时，就是MQA，当ng=nh时，退化成原始的MHA

KV Cache的空间占用率是原先MHA的 ng/nh

GQA还可以将注意力模块的参数量减少到原先使用MHA的 $\eta = 0.5 + 0.5 \cdot ng/nh$ 倍

Some Extensions

*1. **CLA**: If GQA **reuses the KV Cache within layers**, the method proposed by Sun et al. (2024); Brandon et al. (2024); Goldstein et al. (2024) involves inter-layer KV Cache reuse.*

*2. Goldstein et al.(2024) : **generate the all layers' KV Cache using linear models**, like RetNet (Sun et al., 2023a) and RWKV (Peng et al., 2023a).*

*3. DeepSeek-AI et al.(2024) : **Multi-Head Latent Attention (MLA).***
*MLA achieves KV Cache decompression by **utilizing an additional dimension-expanding matrix**, rather than merely reusing the existing content.*

*4. Yen et al. (2024) proposed a new idea for the LLM architecture. CEPE, a framework that combines the pre-trained LLM with an Encoder module that serves as **context compressor.***

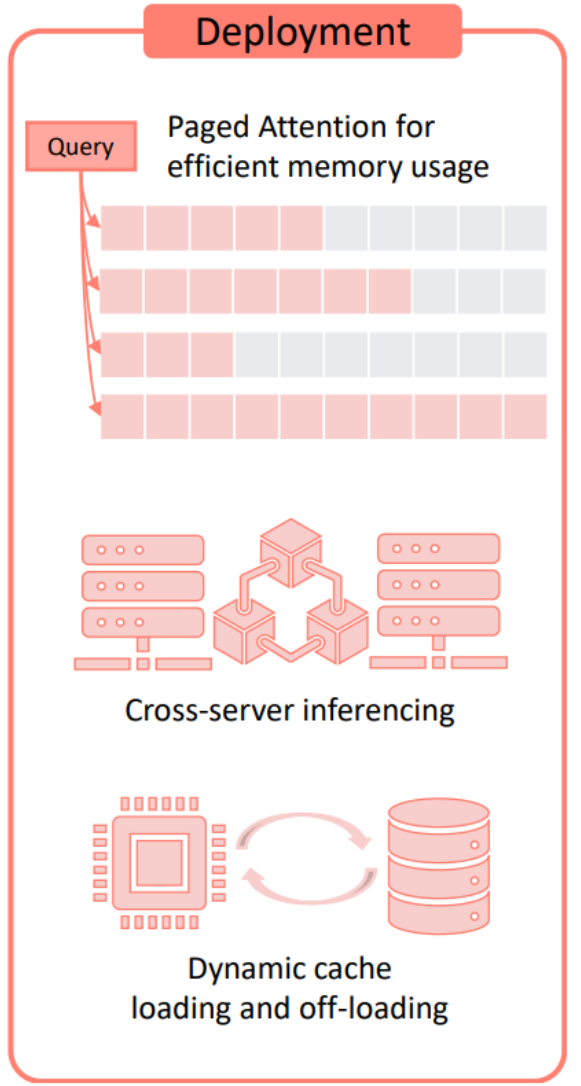
Part 03

Deploy Stage Opts

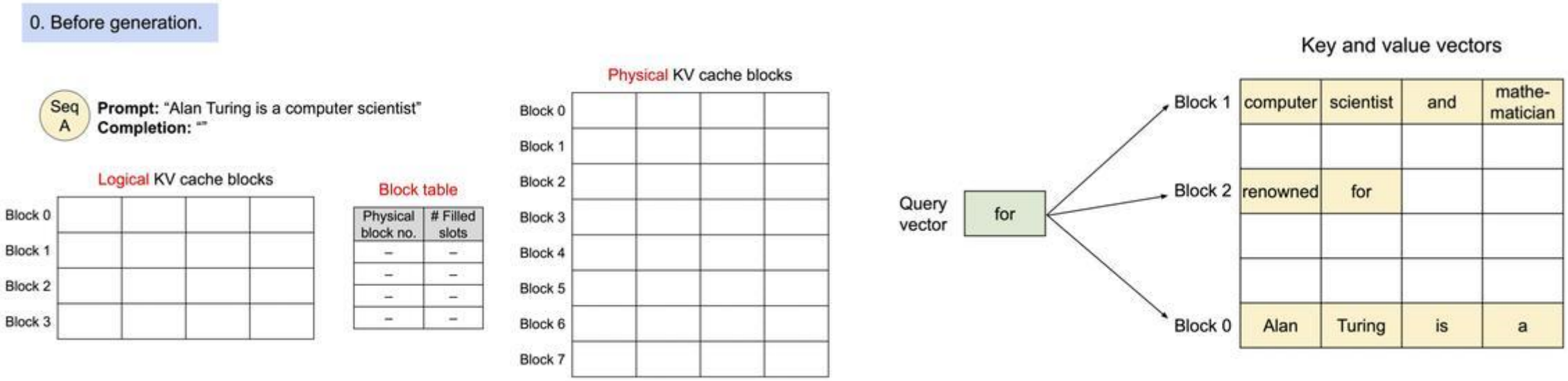
Problems in Deploy Stage

1. 在推理过程中，由于不断有新的token加入KV Cache，导致内存处于反反复复地分配和释放的状态，会导致内存碎片出现。（因为不知道推理完成时的句子长度，所以一般采用预分配max_length最长句子长度的空间，但是一般用不完，导致碎片出现）

2. 在推理过程中，无法批量处理 KV Cache，导致频繁的小批量数据访问占用了大量内存带宽，从而影响计算效率和推理速度。



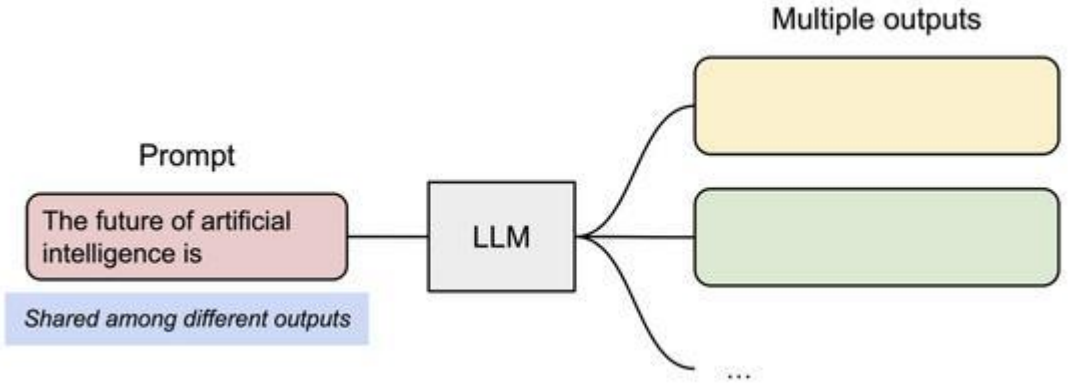
Paged Attention & vLLM

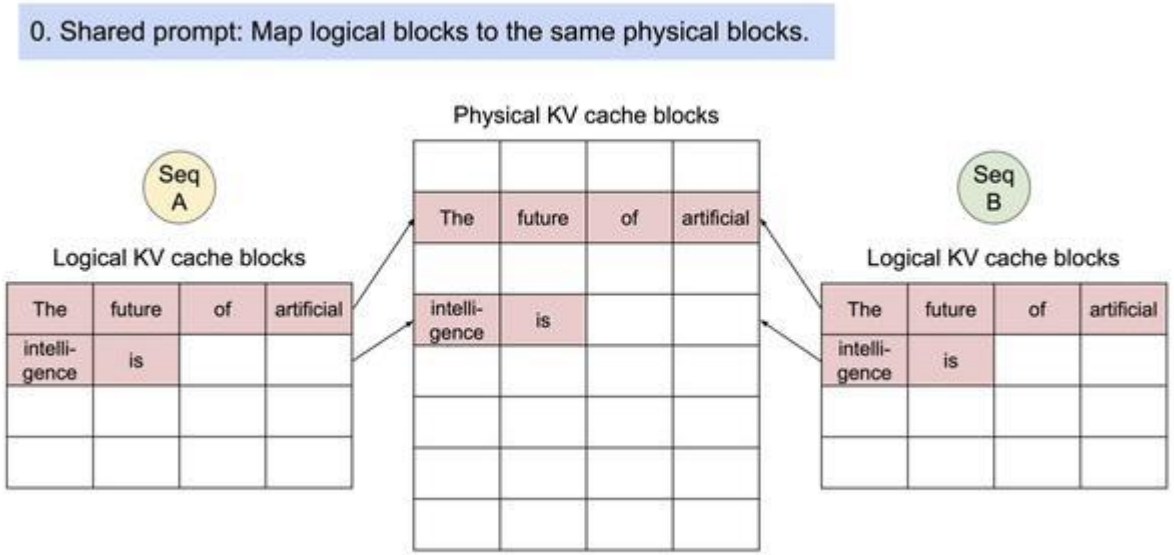


我们在推理的过程中，可以随着KV Cache的增长动态地分配内存，每一次内存分配的单位为一个Block。

每当一个块超出了，就再分配一个新块，每一个Sequence的分配的Block都保存在Block Table中，意味着在内存中分配的块也可以是不连续的。

Paged Attention & vLLM





另外，在对同一个Prompt进行decode采样的时候，解码出来的不同的Sequence的相同的Prompt开头部分可以共享内存空间中的同一块区域。

DistKV-LLM& DistKV-LLM

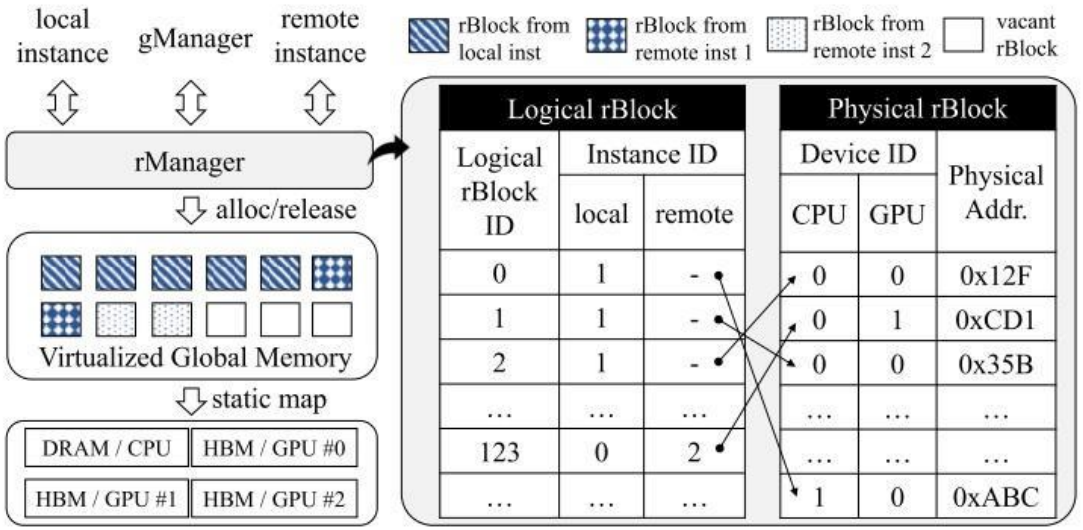


Table 1: LLaMA2-13B, KV Cache size with context length

Context length	10k	100k	500k	1000k
KV Cache size	8.19GB	81.9GB	409.6GB	819.2GB
Misc size	26GB	26GB	26GB	26GB

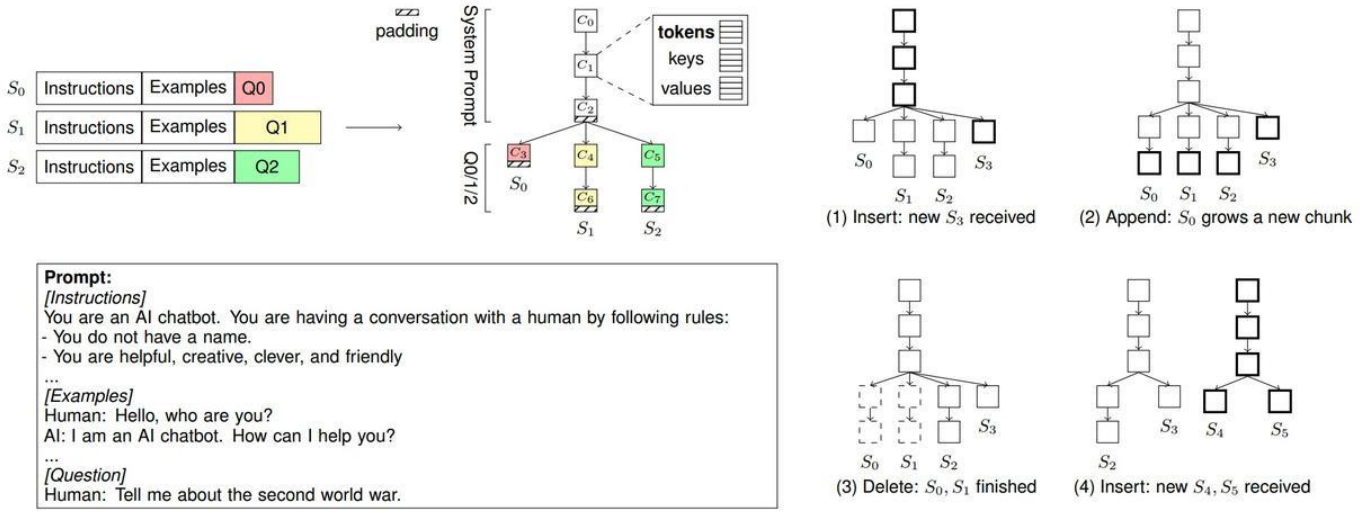
Page_attn的局限性：随着context长度的增加，kv cache内存越来越大，单机已经无法适应需求

DistKV的思想：自己的空间不够，别的节点的空间来凑

每个LLM服务实例都配备有专用的rBlock管理器rManager，维护一个详细的表，该表将这些逻辑rBlock映射到全局内存中对应的物理rBlock地址。

gManager作为一个集中式管理器，维护所有实例的全局内存信息。每个实例周期性地向gManager发送心跳信号，传达关于其剩余可用内存空间的更新。

Chunk Attention



动机：*在处理每个新对话请求时，模型需要进行预填充（pre-fill）阶段，即计算初始的键值对（KV Cache），由于多个请求共享相同的*Prompt，*Prompt前缀中存在显著的重叠*

解决方式：*建立一个字典树（Dictionary Tree），存储所有历史对话的前缀信息。当收到新的对话请求时，通过查找字典树找到最长的公共前缀。对于找到的最长公共前缀，重用对应的KV Cache。*

Some Extensions

1. Jin et al. (2024) and Lee et al. (2024) offload KV Cache to CPU with speculation. The core idea is to offload the primary portion of the KV Cache to the CPU, retaining only critical components.

2. He & Zhai (2024) introduces a novel approach: by collaboratively utilizing the CPU for attention computations, we can enhance GPU utilization efficiency, improve overall effective computational power, and accelerate inference speed.

Part 04

Post-Training Opts

Eviction

Eviction methods are about the policies to discard unnecessary tokens.

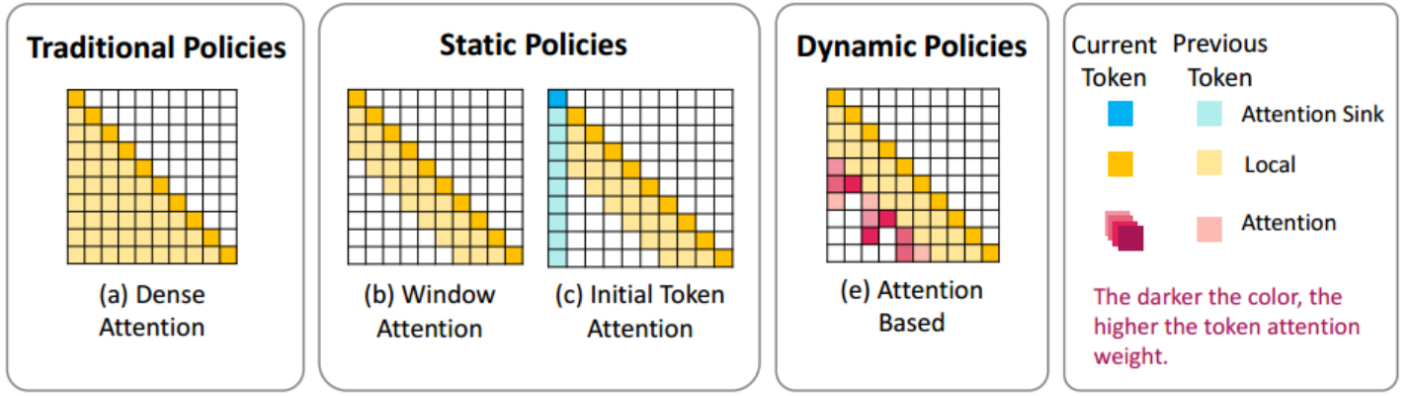
静态规则:

1. *slide window attention (Beltagy et al., 2020).*

2. *Xiao et al. (2023) and Han et al. (2024) proposed keeping KV Cache of both initial and recent tokens.*

动态规则:

注意力权重反映出了单个token的重要性，可以结合attn weights来判断是否丢弃这个token对应的KV Cache .



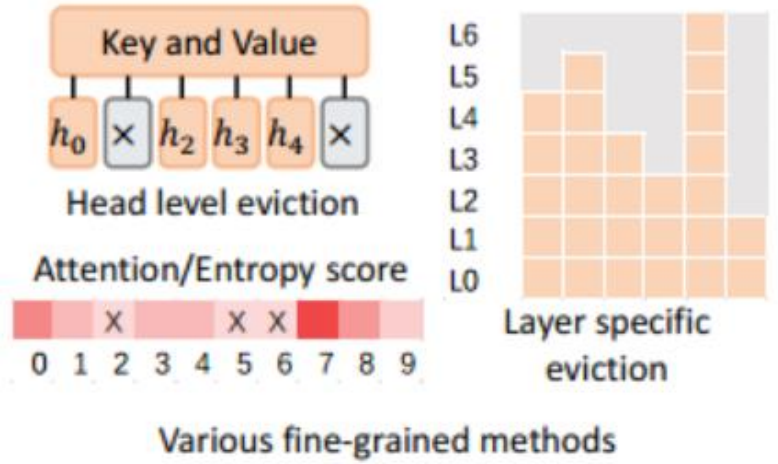
Dynamic Eviction

Repetitive Attention Pattern: a natural hypothesis is that only the tokens which are important in previous steps will be significant in the future.

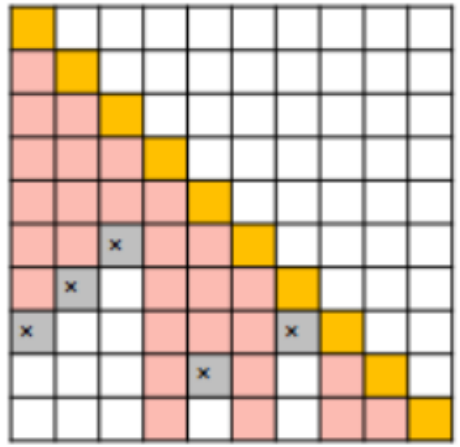
1. Token Omission Via Attention (TOVA) *(Oren et al., 2024) : While keeping a fixed-length of KV-Cahce, TOVA evicts at each decoding step the tokens with minimal attention weights layer-wise.*

2. H2 Eviction Algorithm (Zhang et al., 2023) *uses accumulative normalized attention scores to decide which token to stay and at the same time keeps the recent tokens.*

3. PyramidInfer(Yang et al., 2024a) *uses a layer-wise approach with recent tokens occupying more weights. It defines a decay ratio to shorten the length of KV Cache in deeper layers and thus forming a "pyramid".*



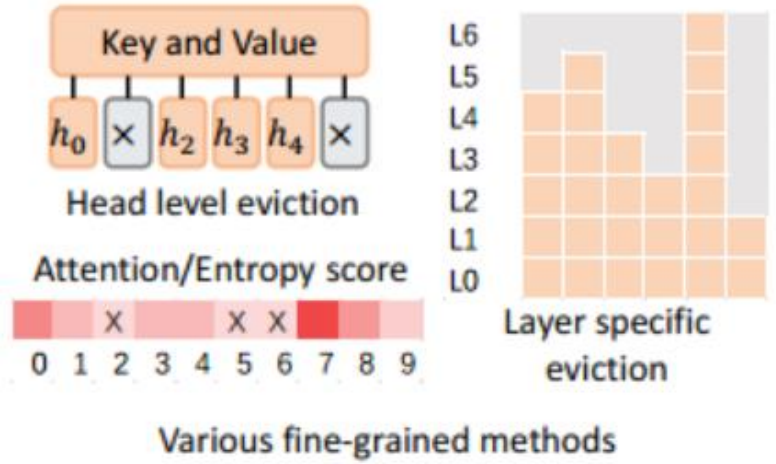
Various fine-grained methods



Eviction helps discard tokens with less information

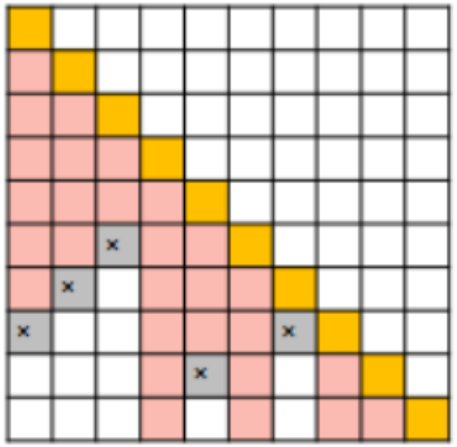
Dynamic Eviction

4. FastGen (Ge et al., 2023) : *a) keeping special tokens, b) keeping punctuation tokens, c) keeping recent tokens and d)keeping tokens with attention-weight-based policies.*



Various fine-grained methods

5. SparQ Attention (Ribar et al., 2023) : *Instead of reducing memory capacity, it aims at reducing the amount of data transferred.*
it keeps a running mean of value vectors and interpolates between this running mean and the output of attention so as to approximate the original distribution.



Eviction helps discard tokens with less information

Merging

1. Nawrot et al. (2024) : *Dynamic Memory Compression (DMC), use a single, after-trained dimension in key to determine whether to merge or not, and use attention score as the reference of merging.*

2. Wang et al. (2024) : *KVMerger, to merge KV by Gaussian weights and attention score.*

3. Yu et al. (2024) *proposed a novel approach to merge KV Cache of each token across different attention head, with a small amount of additional training, eventually turning an MHA model into a GQA model.*

Thanks for your listening!