KV Cache Optimization Methods

- KV cache stands for the key-value tensors computed for past tokens to avoid recomputing them, and is stored and reused during autoregressive decoding.
- KV optimization methods usually happen in these stages:
    - **Pretraining:** examples like MHA, MQA, GQA
        - **Multi-Head Attention (MHA):** Multi-head attention uses heads in parallel with different linear transformations on the queries, keys, values, and outputs. Each head has its own K and V, so MHA stores a separate KV for every head, which is memory-heavy during inference.
        - **Multi-Query Attention (MQA):** Multi-query attention is identical to Multi-Head Attention except that the different heads share a single set of keys and values. It uses multiple heads and one set of keys and values. This reduces the kv cache size by a factor of the number of heads
        - **Group Query Attention (GQA):** In GQA, query heads nh are divided into ng groups. All query heads in a group share the same key and value head. This results in an intermediate number of KV heads between MHA and MQA
    - **Deployment:** examples like PagedAttention, DistAttention, ChunkAttention
        - **PagedAttention:** utilizes the page memory mechanism that maps the KV cache, which used to be stored continuously, to discontinuous GPU memory.
        - **DistAttention:** an attention mechanism that subdivides attention computations across GPUs, to avoid KV cache transfer during decoding.
        - **ChunkAttention:** a prefix-aware self-attention module that can detect matching prompt prefixes across multiple requests and share their key/value tensors in memory at runtime to improve the memory utilization of the KV cache.
    - **Post-training:**
        - **Eviction and merging:**
            - Eviction drops some KV entries (that are less important or contain less information).
            - Merging, an extension of eviction, replaces some KV entries with an aggregated state of keys and values.
        - **Quantization:** compresses the key-value cache by mapping the values from tensors in full precision to discrete levels.
            - **Full quantization:** includes full quantization on the model weights as well as the key-value pairs.
            - **KV quantization:** quantizes the KV activations, selectively targets KV cache activations, and doesn't affect model weights.

References:

1. Shi, L. *et al.* (2024) *Keep the cost down: A review on methods to optimize LLM' s KV-cache consumption*, *arXiv.org*. Available at: https://arxiv.org/abs/2407.18003 (Accessed: 10 February 2026).
2. Noam Shazeer.Fast transformer decoding: One write-head is all you need.CoRR, abs/1911.02150, 2019.URL http://arxiv.org/abs/1911.02150.