# Dynamic Sparsification and Comparative Analysis of KV Cache Management in LLM Inferrence.

Oteo Mamo[1], Olga Kogiou, Trivikram Dharmavarapu , Weikuan Yu[1]
Florida State University

## Overview

### KV Cache Management

- Addressing memory and performance challenges in KV cache for efficient LLM inference.

### Current Challenges

- Memory pressure from KV caches growing linearly with sequence length, layers, and concurrent requests, often exceeding model parameters.
- Inefficiencies in trade-offs between latency, throughput, and memory under varying model sizes and workloads.

### Project Goals

- Empirically compare vLLM, InfiniGen, FlexGen+H2O across metrics like TTFT, latency, memory.
- Identify bottlenecks and provide recommendations for selection under constraints.
- Examine sparsity impact on memory efficiency and accuracy.

## Introduction

LLMs rely on Key-Value caching to accelerate inference, but this comes with significant memory overhead that grows with sequence length, model depth, and request scale.

$$Mem_{KV} = D \times H \times (L_0 + t) \times d_h \times \text{sizeof(dtype)} \times 2$$

vLLM, InfiniGen, FlexGen, and FlexGen+H2O offer distinct strategies such as sparse attention, memory offloading, and KV compression to manage the KV cache during inference.

| Feature | vLLM | InfiniGen | FlexGen + H2O |
|---|---|---|---|
| Hierarchical Placement | GPU | CPU/GPU | CPU+Disk |
| Sparsification | ✗ | Block-level | Token-level |
| Quantization | ✗ | INT8 | INT4 |
| Cache Granularity | Paged Attention | Block-level | Paged blocks |
| Eviction Policy | LRU | FIFO-like | Aggressive |
| Paging | ✗ | ✓ | ✓ |
| KV Cache Utilization | Full | <10% | ~20% |
| Batching Strategy | Prefix sharing | Static | Static |

vLLM excels in high-throughput scenarios, InfiniGen offers faster response with lower memory, and H2O with FlexGen minimizes memory at a cost to performance.

## Research Questions

**Resource-Constrained Selection:** Given limited GPU memory and varying throughput or latency demands, which framework offers the best trade-off for a particular deployment scenario?
**Scalability and Memory Behavior:** How does memory usage evolve for each framework at scale—under increasing model sizes, request concurrency, and longer prompt or output sequences?
**Impact of Sparse KV Caches:** How do techniques such as attention sparsification affect memory efficiency and inference accuracy?

## Research Findings

The performance differences across KV cache frameworks stem from how each system manages memory locality, cache prefetching, and GPU-CPU communication. **vLLM** performs best at scale because it keeps the entire KV cache on GPU and minimizes memory ragmentation using preallocated tiled blocks. **InfiniGen** prefetches only the most relevant KV entries dynamically, which reduces memory usage but eventually hits limits as working sets grow. **FlexGen** and **FlexGen+H2O** offload most KV data to CPU and apply quantization, trading latency for the ability to run on low-memory hardware.

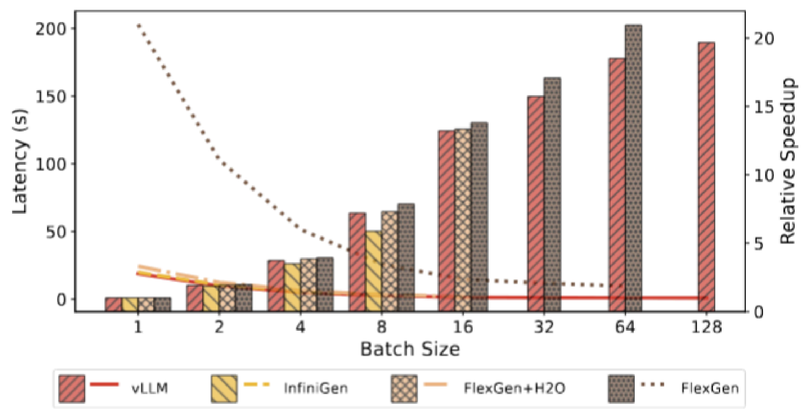


Fig. 3: Relative Speedup (columns) and latency (lines) for OPT-13B with token configuration of 512x512.

**Observations:**

- Architectural trade-offs arise from how each system handles KV cache allocation, eviction, and reuse under memory pressure.
- Frameworks optimize differently for compute locality vs. memory hierarchy traversal (e.g., GPU-resident vs. CPU-offloaded caches).
- As sequence length and model size grow, efficient KV cache tiling, streaming, and sparsity become essential to sustain inference performance.

**Attention sparsification** works effectively because many tokens in long contexts contribute little to future predictions—so selectively caching top-k keys balances memory reduction with semantic fidelity. By dynamically selecting and caching only the most salient key-value pairs, systems can significantly reduce memory and latency while preserving accuracy on most reasoning tasks.

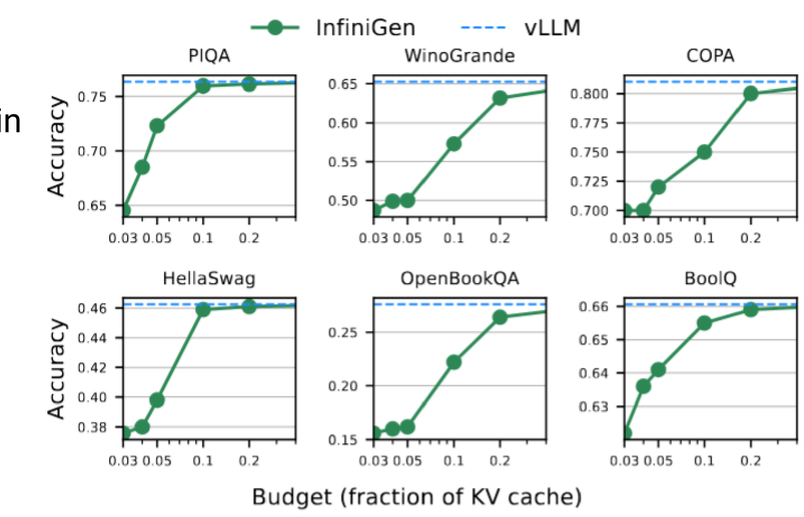


Fig. 5: Accuracy comparison of InfiniGen and vLLM across PIQA, WinoGrande, COPA, HellaSwag, OpenBookQA, and BoolQ datasets.

## Conclusion

- Resource-Constrained Settings: InfiniGen offers the best balance of low latency and moderate memory. H2O supports higher concurrency but incurs greater latency; hierarchical memory and prefetching help mitigate this.
- Scalability and Memory: All systems scale with model size, but performance degrades with high concurrency. vLLM maintains efficiency at scale using paged attention and prefix-sharing.
- Sparsification Benefits: Top-k sparsification reduces KV cache memory with minimal accuracy loss, offering a flexible trade-off between memory footprint and inference quality.

## Future Work

Develop adaptive hybrid policies focused on the sparsity of LLMs (k-top token memory reduction, MHA layer reduction).

## Acknowledgement and Contact

1. This work is supported in part by the National Science Foundation awards 1763547 and 2403089, and has used the AWS services funded by a grant by the Florida State University and the NoleLand facility that is funded by the U.S. National Science Foundation grant CNS-1822737.
2. Contact: om21d@fsu.edu, (850) 518-9740