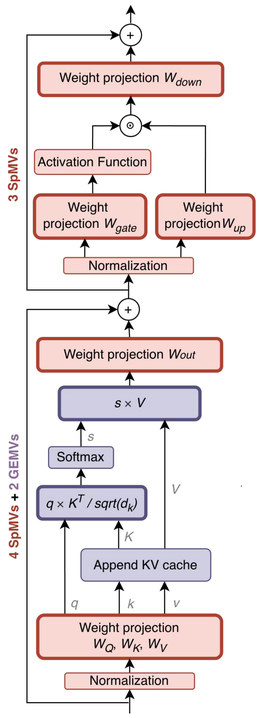
Leveraging Sparsity for LLM Inference in a Commercial Processing-In-Memory Prototype

Irene Simó Muñoz, Giulia Guidi
isimo@cs.cornell.edu
Cornell Bowers College of Computing + Information Science
ALPS Lab

Why

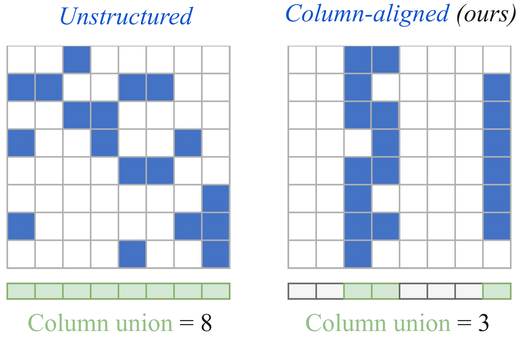
- Decode produces **one token at a time**, so every LLM layer boils down to a matrix–vector product. No data reuse: **time per token is dominated by reading data** (weights and the KV cache) **from memory.**
- Bank-level PIM (SK hynix AiM) attacks this directly by **computing next to the data**. Matrix stays resident in DRAM with SIMD, which **suits dense GEMV well**.
- One shared command stream forces every bank to read the same column index. After pruning, zeros across columns are still used, so on this hardware **standard pruning saves nothing at all**



Design space characterization

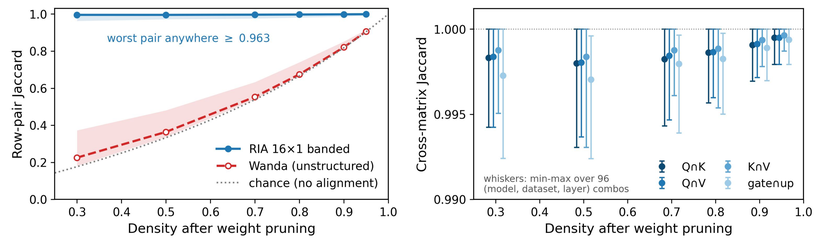
- Row i of a lockstep bank group keeps columns C_i. The group must visit every column in the union $U = \text{union}(C_i)$
- The real cost is therefore $|U|$, not the nonzero count.

Cost vanishes only if all rows keep the same columns



Mechanism 1: Weight pruning

- **Activation importance is a column property**: ~0.1% of hidden dimensions carry 10-100x the median norm



Mechanism 2: KV cache sparse attention

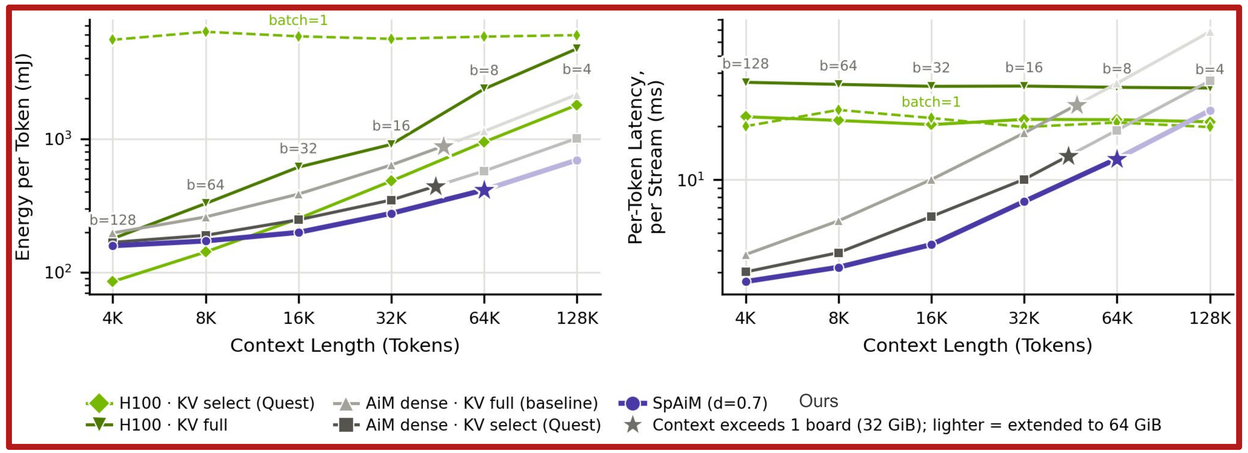
- Weight pruning does not help the attention scan, which grows with context length and eventually dominates.
  - Quest fixes this on GPUs by **reading only the tokens each query cares about while keeping the full KV cache**, but it picks them per head.
- We reuse the trick from the projections:
  - Heads that share a command agree on a single token set per GQA group. The union shrinks back to the 8% budget and attention cost per token stays roughly constant as context grows, with no loss in retrieval accuracy (Results table below)

Results

2.3 - 3.1x
Less energy than dense full-KV PIM

1.7 - 2.6x
Less energy than the measured H100 @ 8%

2.4x
Higher throughput than dense full-KV



Up to 1.35x
Larger context stored on-device

Weight pruning reduces the numbers of boards needed for large mdoels

Accuracy

Ctx	full KV (100%)	Window (25%)	Quest [2] (8%)	Ours (8%)
4K	0.943 / 0.885	0.064 / 0.075	0.942 / 0.867	0.940 / 0.858
8K	0.940 / 0.815	0.086 / 0.083	0.938 / 0.822	0.938 / 0.812
16K	0.950 / 0.797	0.068 / 0.077	0.950 / 0.820	0.948 / 0.823
32K	0.935 / 0.790	0.082 / 0.083	0.937 / 0.783	0.935 / 0.748
64K	0.887 / 0.765	0.073 / 0.073	0.887 / 0.772	0.872 / 0.757

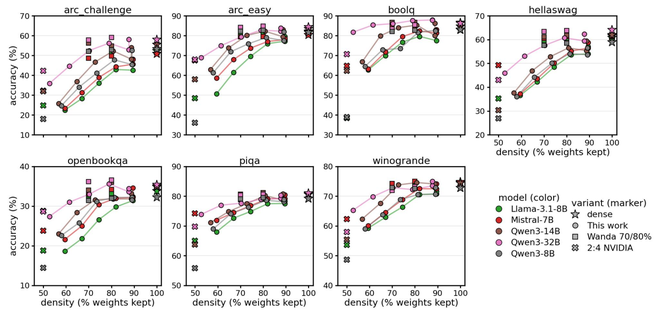
RULER [4] retrieval accuracy over increasing context lengths, ↑is better (Qwen3-14B / Mistral-7B).

Weight pruning

Dense model accuracy kept up to 30% pruning

KV cache sparse attention

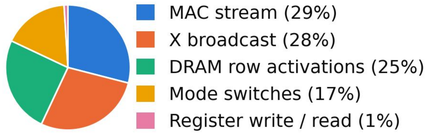
Adapted token selector keeps GPU (native) retrieval accuracy for long contexts



Future work

Compose sparsity and quantization: Pruning removes weights and quantization represents them cheaply, compounding the memoryfootprint and data movement savings

Explore beyond stock hardware: GEMV execution in stock AiM is still bottlenecked by several overheads. Architectural modifications can serve as mechanisms to directly eliminate these overheads.



[1] RIA: Zhang et al., ICLR'24 [2] Quest: Tang et al., ICML'24 [3] AiM: Kwon et al., ISSCC'22 [4] RULER: Hsieh et al., COLM'24 [5] Ramulator2: Luo et al., CAL'24 [6] SparseGPT: Frantar & Alistarh, ICML'23

Acknowledgments to Cecilio C. Tamarit, Joseph Maheshe, Mukund Gaur and José Martínez