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

Irene Simó Muñoz✉
isimo@cs.cornell.edu
Cornell University
Ithaca, New York, USA

Giulia Guidi
gguidi@cornell.edu
Cornell University
Ithaca, New York, USA

Abstract

Large language model decoding is bounded by the memory wall: weights and the KV cache must be moved every token, and the low-arithmetic-intensity matrix–vector products of decode leave throughput set by bandwidth rather than compute. Bank-level processing-in-memory (PIM) attacks this directly by keeping weights resident and computing in place, but its lockstep, all-bank execution (every bank issuing the same command against a shared operand) provides no way to skip the zeros that pruning produces. We present the first end-to-end sparsity-aware inference framework for bank-level PIM. A *column-union* cost model shows that lockstep cost is set by the union of columns (or KV tokens) a bank-group must jointly visit, not by the nonzero count, and we drive that union to its minimum on both halves of the decode layer. Activation-aware weight pruning [10] makes surviving columns align across rows and across matrices for free (the hardware-optimal mask and the accuracy-preserving mask coincide) while a group-aligned adaptation of query-aware KV selection [8] collapses the attention scan that pruning cannot touch. Against a measured NVIDIA H200 at batch-1, this work cuts single-stream per-token latency by ~4–7× and, beyond moderate context, undercuts even the GPU's most energy-efficient batched configuration (crossing over near 10K, ~6× lower at 128K), while preserving downstream accuracy and extending on-board context.

1 Introduction

Serving LLM inference is a data-movement problem. Decoding emits one token at a time, so each layer collapses to matrix–vector products with almost no reuse: performance is set by how fast weights and the KV cache stream, not by compute, while the resident weights and growing cache squeeze capacity. Bank-level SIMD PIM (SK hynix AiM [4], Samsung Aquabolt [5]) fits this GEMV-shaped workload exactly: each bank streams a weight row against a broadcast activation, all banks firing one operation in lockstep.

But PIM only removes the movement of *dense* data, and the feature that makes it efficient fights compression: all banks share one command stream, so every bank must address the *same* column index each cycle. A pruned matrix wastes cycles on empty columns; a naively compressed one breaks lockstep. Unlike a GPU, where pruning pays once irregularity is tamed, on lockstep PIM the saving vanishes outright.

This work is, to our knowledge, the first end-to-end sparsity-aware inference framework for bank-level PIM. We co-design sparsity with the lockstep model across *both* halves of the decode layer:

- **Projections.** A column-union cost model shows lockstep cost is set by the union of columns a bank-group jointly visits; activation-aware pruning (RIA [10]) drives that union to its density-minimum for free.
- **Attention.** We adapt query-aware KV token selection (Quest [8]) to the shared-operand constraint, aligning each group's selected tokens so one broadcast serves a shared set, taming the attention scan pruning cannot touch.

2 Sparsity in Processing-in-Memory

A group of banks advances in lockstep under a single command and reads a shared broadcast operand: every bank addresses the same column index within its own row each cycle. Let C_i be the columns surviving in row i of the group. The group must visit every column in the union $U = \bigcup_i C_i$, regardless of how many rows use it:

$$\text{cost(group)} \;\propto\; |U|, \qquad \text{not} \qquad \textstyle\sum_i |C_i|.$$

The engine pays the union, not the mean occupancy; the gap is a pure alignment tax, vanishing only when the C_i coincide. The objective is thus to minimize the per-group column union at a quality budget, not to minimize nonzeros.

2.1 Pruning the projections

We recover the alignment from the model rather than engineering a mask. Activation importance is a *per-column* quantity (a tiny fraction of hidden dimensions carry norms 10–100× the median [9]), the *same* columns for every row. Unstructured magnitude pruning [7] lets each row's own weights decide, and survivors scatter to chance-level overlap (Fig. 1). RIA [10] normalizes magnitude by row/column statistics at a block granularity matched to the group (16×1), so the shared column-activation term dominates: the worst row-pair Jaccard stays ≥ 0.963 across every model, dataset, layer, and density (Fig. 1). The mechanism reaches across matrices: projections sharing an input (W_Q, W_K, W_V; $W_{\text{gate}}, W_{\text{up}}$) prune to nearly the same columns (Jaccard ≈0.998, Fig. 2), letting us batch them into one schedule at no extra column cost. A per-block least-squares reconstruction [1] recovers accuracy; the partnerless, perturbation-sensitive projections (W_O, W_{down}) stay dense, since alignment cannot help them.

2.2 Pruning the attention

Weight pruning leaves the attention scan ($QK^\top$, SV), which grows with context and eventually dominates—the attention wall. Query-aware KV selection (Quest [8]) keeps only each query's most relevant tokens, but the lockstep tax reappears in KV tokens: vanilla Quest selects per head, so a shared broadcast must serve the union of all heads' selections—2.6× the nominal budget, touching 89% of DRAM rows. The fix mirrors the projections: heads sharing a command agree on one token set per group (group-aligned selection),

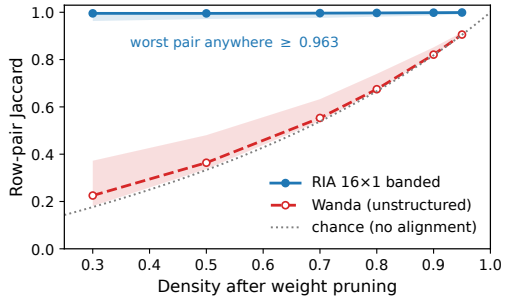


Figure 1: Within-matrix alignment is a property of the metric, not an imposed mask. Jaccard overlap between two rows' surviving column sets. RIA-16×1 keeps rows on the same columns at every density (worst pair ≥ 0.963); unstructured Wanda tracks the no-alignment chance line. Models: Mistral-7B-v0.1, Qwen3-8B/14B/32B.

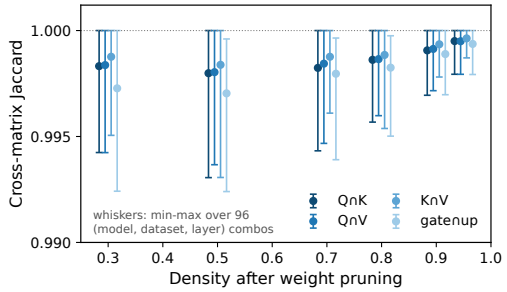


Figure 2: Alignment extends across matrices that share an input. Jaccard overlap of surviving columns between $Q/K/V$ and between gate/up, ≈ 0.998 across densities; whiskers span min–max over 96 (model, dataset, layer) combinations. The shared activation is what makes cross-matrix batching free.

and the union collapses back to the budget (8%), keeping per-token attention cost flat (Fig. 4). Composed with weight pruning, retrieval is preserved: RULER [3] tracks full KV within half a point through 32K and 2.6 points at 64K, matching per-head selection at the same budget.

3 Results and Discussion

We use cycle-accurate, trace-driven simulation [6] with the PIM command set and a command-activity energy model from a validated PIM system [2]. GPU baselines are measured on an NVIDIA H200.

Accuracy. Across seven zero-shot tasks and eight models (7B–32B), RIA-16×1 tracks the dense baseline and unstructured Wanda [7] down to 70–80% density, stays competitive with NVIDIA 2:4, and matches or beats INT4 quantization at equal footprint (Fig. 3)—column-aligned pruning pays no accuracy premium.

Gains over dense PIM.. In Fig. 4 (Qwen3-14B, batch-1), dense full-KV energy climbs steeply with context, and per-head selection gives back much of its saving to the group union. Group-aligned 8% selection with d=0.7 pruning stays nearly flat: 1.3–1.5× below per-head selection and 2.3–3.1× below dense full-KV at 32K–128K. Boards are weight-dominated, so freed capacity converts into up to ∼2.5× longer resident context or fewer boards.

Gains over the GPU.. Against the measured H200 (Qwen3-14B) we compare energy at the GPU's *most efficient* configuration—fused token selection (8%) at its capacity-optimal batch (Fig. 4). Below ∼10K the heavily batched GPU wins, but its optimal batch shrinks as KV fills HBM, so its per-token energy climbs while ours stays flat (resident weights). Ours crosses near 10K and pulls steadily ahead—1.7× lower at 32K, 2.3× at 64K, and 2.6× at 128K (3.6× at a 3.1% budget)—at batch-1, whereas the GPU needs dozens of concurrent requests to approach even that. Our regime is single-stream, long-context serving—the on-device, low-concurrency operating point.

Relevance to interactive and low-power inference. This low concurrency regime is particularly relevant to interactive and on-device inference, where a single user generates a latency-sensitive stream of decode requests rather than a datacenter-scale batch. Mobile, edge, and other power-constrained platforms cannot amortize memory traffic across dozens of concurrent users, making the energy and capacity cost of moving weights and KV state especially important. Our design targets precisely this operating point: weights remain resident in PIM, while column- and token-aligned sparsity reduces the amount of data that must be accessed per generated token. The result is a system-level tradeoff that is difficult to obtain from throughput-oriented accelerators alone: long-context, single-user inference with bounded per-token energy and substantially reduced memory footprint. These properties make bank-level PIM particularly attractive as inference moves closer to the user and away from highly provisioned, highly batched datacenter serving.

4 Conclusions

PIM answers the memory wall by keeping weights resident, but only sparsity removes the work that remains—and only if it respects lockstep. Our column-union cost model exposes the one quantity governing both halves of the layer: the columns (weights) or tokens (KV) a lockstep group must jointly visit. Activation-aware pruning minimizes the weight-column union for free; group-aligned token selection does the same for KV tokens. The lockstep execution that appears to forbid sparsity is exactly what makes it efficient, once format, kernel, and selection policy are co-designed around the per-group union. The poster walks the cost model, both mechanisms, and the sweeps behind each claim.

References

[1] Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot. In *International Conference on Machine Learning (ICML)*, Vol. 202. 10323–10337.

[2] Yufeng Gu, Alireza Khadem, Sumanth Umesh, Ning Liang, Xavier Servot, Onur Mutlu, Ravi Iyer, and Reetuparna Das. 2025. PIM Is All You Need: A CXL-Enabled GPU-Free System for Large Language Model Inference. In *Proc. 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 862–881. doi:10.1145/3676641.3716267

[3] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. RULER: What's the Real Context Size of Your Long-Context Language Models?. In *Conference on Language Modeling (COLM)*.

[4] Hyucksung Kwon et al. 2022. A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level

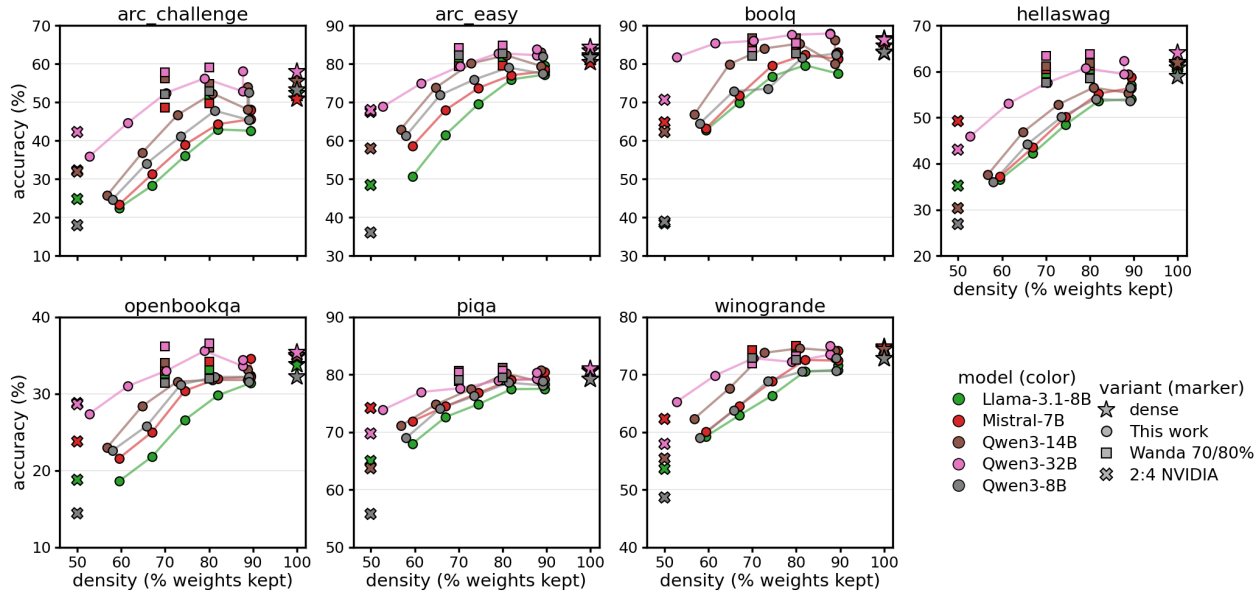


Figure 3: Downstream accuracy vs. density. This work – RIA-16×1 – (circles) tracks dense (stars) and Wanda (squares), and matches or beats 2:4 at equal footprint, across 7 tasks and 8 models.

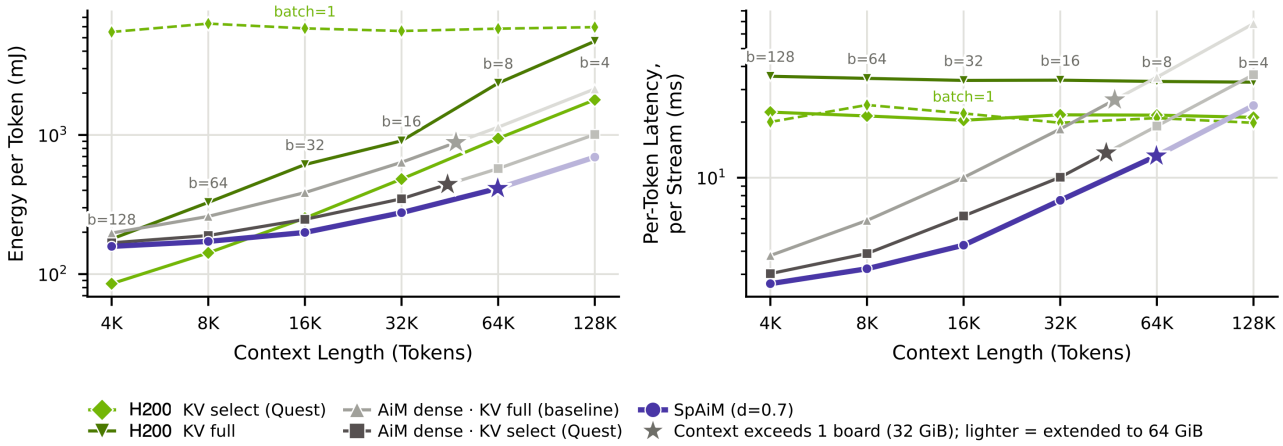


Figure 4: End-to-end energy per token and per-token latency as context length increases (Qwen3-14B). One H200 GPU vs. One AiM board.

Parallelism, for Machine Learning Applications. In *Proc. IEEE International Solid-State Circuits Conference (ISSCC)*.

[5] Young-Cheon Kwon, Suk Han Lee, Jaehoon Lee, Sang-Hyuk Kwon, Je Min Ryu, Jong-Pil Son, et al. 2021. A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 64. 350–352. doi:10.1109/ISSCC42613.2021.9365862

[6] Haocong Luo, Yahya Can Tuğrul, F. Nisa Bostancı, Ataberk Olgun, A. Giray Yağlıkçı, and Onur Mutlu. 2024. Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator. *IEEE Computer Architecture Letters* 23, 1 (2024), 112–116. doi:10.1109/LCA.2023.3333759

[7] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A Simple and Effective Pruning Approach for Large Language Models. In *International Conference on Learning Representations (ICLR)*.

[8] Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. QUEST: Query-Aware Sparsity for Efficient Long-Context LLM Inference. In *International Conference on Machine Learning (ICML)*.

[9] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*. PMLR, 38087–38099.

[10] Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. 2024. Plug-and-Play: An Efficient Post-training Pruning Method for Large Language Models. In *International Conference on Learning Representations (ICLR)*.