Late Breaking Results:

Polaris: Enabling PIM-Optimized Low-Energy SpGEMM using a Remote-Access-Aware Mapping Solution

Helya Hosseini
University of Maryland, College Park

Christina Giannoula
Max Planck Institute for Software Systems (MPI-SWS)

Bahar Asgari
University of Maryland, College Park

Abstract

Sparse kernels are widely used in applications ranging from scientific computing to machine learning. Among them, SpGEMM is particularly challenging due to irregular memory accesses, making it highly memory-bound and dominated by data movement. Processing-in-Memory architectures can mitigate this cost by performing computation within memory banks, but irregular sparsity often causes load imbalance and costly cross-bank accesses. To address these challenges, we propose **Polaris**, a PIM-aware mapping solution for efficient SpGEMM on HBM-based systems. Polaris leverages a row-wise-product formulation and a sparsity-aware mapping strategy to improve locality and balance workloads across banks, reducing data movement and achieving on average 2× lower energy consumption and up to 8× speedup over prior designs.

1 Introduction

Sparse kernels such as sparse matrix–vector multiplication (SpMV) and sparse general matrix–matrix multiplication (SpGEMM) are widely used in scientific computing and machine learning. Many accelerators have been proposed to improve sparse computation and memory efficiency [1, 4, 5, 9]. However, SpGEMM remains challenging due to irregular, data-dependent memory accesses that make it highly memory-bound. Although prior architectures optimize dataflow and compute pipelines [7], they do not substantially reduce irregular memory accesses, leaving data movement as the dominant performance and energy bottleneck.

Processing-in-Memory (PIM) offers a promising solution by performing computation directly within DRAM banks, reducing costly data transfers and exploiting the high internal bandwidth of stacked memories such as HBM. However, efficiently executing sparse workloads on bank-level PIM cores is difficult because operands must be distributed across banks for local execution. Irregular sparsity further causes load imbalance and cross-bank operand accesses, which are expensive and often unsupported by bank-local PIM designs.

Prior work has explored PIM-based accelerators to reduce data movement for sparse workloads [4, 8]. While these approaches demonstrate performance and energy improvements, they mainly target SpMV or SpMM and often rely on data replication to avoid cross-bank dependencies. This introduces memory overhead and still incurs redundant data movement. Consequently, achieving energy-efficient and well-balanced SpGEMM execution on PIM architectures remains challenging, motivating more adaptive mapping strategies.

To address these challenges, we propose **Polaris**, a mapping solution that co-optimizes data placement and near-memory computation for efficient SpGEMM on HBM-based PIM systems. Polaris adopts a row-wise-product formulation that exposes per-row locality and avoids global intermediate reductions, making it well suited for PIM architectures without remote bank access. Polaris introduces a *PIM-aware mapping strategy* that exploits row-dependency scores and sparsity-aware grouping to place A- and B-rows across banks, improving locality, reducing replication, and balancing workloads. Our evaluation shows that Polaris significantly reduces data movement and achieves on average 2× lower energy consumption compared to prior designs.

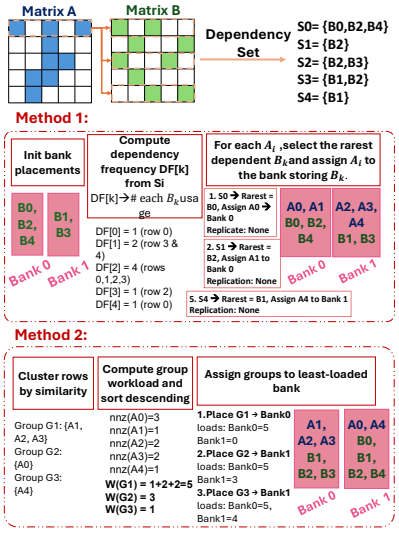

Figure 1: Method 1 assigns each A-row based on its rarest dependent B-row. Method 2 clusters rows with similar dependencies and maps groups to banks.

2 Polaris

The challenge of running SpGEMM on bank-local PIM is that sparsity creates irregular dependencies between rows of A and B, leading to load imbalance, replication of frequently accessed rows, and costly cross-bank data movement. Since bank-level PIM cores cannot perform remote fetches, operands required for a computation must be co-located within the same bank while maintaining balanced workloads. The row-wise SpGEMM formulation naturally exposes *dependency sets* between each A_i and the B-rows it accesses. By analyzing these sets, rows can be mapped to banks to maximize locality, reduce replication, and balance computation. *Polaris leverages this observation through two sparsity-aware mapping strategies that trade off replication, preprocessing cost, and load balance.*

Mapping Strategy Overview Polaris implements two sparsity-aware mapping methods that use the SpGEMM dependency structure to place A- and B-rows across banks (Figure 1). Both methods

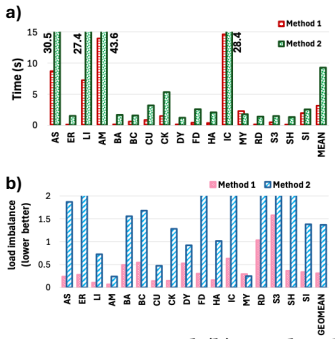

Figure 2: (a) Mapping time and (b) Load imbalance across banks for the two methods.

execute each A_i in a bank containing its required B-rows, minimizing cross-bank accesses and redundant data movement.

Method 1: Dependency-Frequency Mapping As shown in Figure 1, Method 1 assigns each A_i to the bank containing the *rarest* B-row in its dependency set S_i. The algorithm first computes the frequency of each B-row, then anchors A_i to the bank storing the least frequent dependency, replicating any remaining required B-rows if necessary. Anchoring placement to rare dependencies minimizes costly replication while maintaining good load balance, and the greedy row-wise strategy keeps preprocessing overhead low.

Method 2: Group-Based Sparsity-Aware Mapping It clusters A-rows with similar dependency sets and assigns each cluster to a bank. Instead of replicating rows per A_i, the algorithm replicates only the union of the cluster's required B-rows, reducing redundant copies. However, clustering increases preprocessing overhead and may introduce load imbalance when large groups concentrate work on fewer banks. The two methods provide complementary trade-offs: Method 1 prioritizes balance and low preprocessing cost, while Method 2 reduces replication. Polaris adopts Method 1 for the final evaluation due to its better balance and faster mapping time.

3 Experimental Setup

Evaluation Methodology We compare Polaris with HPN-SpGEMM [2], the only PIM-based design tailored for SpGEMM in HBM systems, using identical HBM timing and organization parameters for fairness. We extend the event-driven AttAcc simulator [6] to model bank-level PIM operations, HBM timing, and data movement. Both mapping methods are implemented on an Intel Core i9-14900K to generate placement schedules prior to simulation.

Workloads and Metrics We evaluate Polaris on diverse SuiteSparse matrices [3] representing scientific, structural, and graph workloads with extreme sparsity ($\geq$ 99%). We measure mapping quality (load balance and preprocessing time), performance (execution cycles), and energy using the AttAcc/FGDRAM model (0.55 pJ/b bank-local, 1.56 pJ/b bank-group, 0.5 pJ/b TSV transfer, 0.32 pJ/b MAC). All improvements are reported relative to HPN-SpGEMM.

4 Evaluation

Mapping Time. As shown in Figure 2a, Method 1 incurs 2.9× lower preprocessing time. Its row-wise greedy assignment requires only simple dependency-frequency lookups, whereas Method 2 requires computing signatures, clustering, and group scheduling.

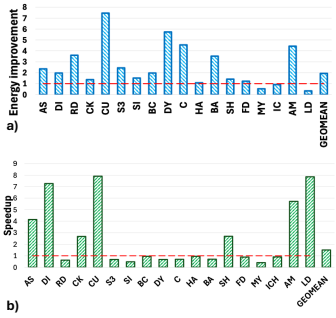

Figure 3: a) Energy improvement and b) speedup of Polaris over HPN-SpGEMM.

Load Imbalance. Figure 2b shows the imbalance factor (std. dev. over mean) across banks. Method 1 achieves on average 4.4× lower imbalance than Method 2, reflecting more uniform distribution of multiplication work. These results indicate that Method 1 provides a better balance between mapping efficiency and workload distribution, achieving both lower preprocessing overhead.

Energy Consumption. Figure 3a shows Polaris' energy savings relative to HPN-SpGEMM. Polaris reduces total energy by an average of 2× across matrices, achieving up to 7× improvement on workloads with highly irregular sparsity such as *CurlCurl_3(CU)*. These benefits stem from Polaris' remote-access-aware mapping, which co-locates all required A and B rows within each bank and eliminates the baseline's repeated broadcasts and cross-bank-group transfers. For small or lightly populated matrices (e.g., *mycielskian13(MY)*), HPN-SpGEMM may match or slightly outperform Polaris due to fewer broadcasts and lower fixed overheads.

Speedup. Figure 3b reports speedup over HPN-SpGEMM. Although Polaris primarily targets energy efficiency, its mapping-driven locality improves performance as well. For large matrices with more NNZ and irregular access patterns (e.g., *CurlCurl_3(CU)*, *ldoor(LD)*), Polaris achieves up to 8× speedup by avoiding cross-bank-group traffic and enabling near-ideal parallel execution. A few small-NNZ matrices exhibit slight slowdowns because Polaris' bank-level coordination overhead does not amortize on tiny workloads.

Acknowledgment

We acknowledge the support of NSF PPoSS Award Number 2316177.

References

[1] U. Bakhtiar et al. 2025. Chasoň: Supporting Cross HBM Channel Data Migration to Enable Efficient Sparse Algebraic Acceleration. In *MICRO*. 778–794.
[2] H. Chen et al. 2024. HPN-SpGEMM: Hybrid PIM-NMP for Sparse General Matrix-Matrix Multiplication. In *CAL*.
[3] T. Davis et al. 2011. The University of Florida Sparse Matrix Collection. *TOMS* 38, 1 (2011), 1.
[4] C. Giannoula et al. 2022. SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures. *Proc. ACM Meas. Anal. Comput. Syst.* 6, 1, Article 21 (2022).
[5] D. Joo et al. 2025. Coruscant: Co-Designing GPU Kernel and Sparse Tensor Core to Advocate Unstructured Sparsity in Efficient LLM Inference. In *MICRO*. 232–245.
[6] J. Park et al. 2024. AttAcc! Unleashing the Power of PIM for Batched Transformer-Based Generative Model Inference. In *ASPLOS*. 103–119.
[7] N. Srivastava et al. 2020. MatRaptor: A Sparse-Sparse Matrix Multiplication Accelerator Based on Row-Wise Product. In *MICRO*. 766–780.
[8] X. Xie et al. 2021. SpaceA: Sparse Matrix Vector Multiplication on Processing-in-Memory Accelerator. In *HPCA*. 570–583.
[9] S. Yadav et al. 2025. MISAM: Machine Learning Assisted Dataflow Selection in Accelerators for Sparse Matrix Multiplication. In *MICRO*. 824–838.