SparseP: Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures

Christina Giannoula[‡†] Ivan Fernandez[‡§] Juan Gómez-Luna[‡]
Nectarios Koziris[†] Georgios Goumas[†] Onur Mutlu[‡]
[‡]*ETH Zürich* [†]*National Technical University of Athens* [§]*University of Malaga*

Sparse Matrix Vector Multiplication (SpMV) is one of the most thoroughly studied scientific computation kernels, because it lies at the heart of many important applications from the scientific computing, machine learning, and graph analytics domains. SpMV performs indirect memory references as a result of storing the sparse matrix in a compressed format, and irregular memory accesses to the input vector due to the sparsity pattern of the input matrix [1–3]. Thus, in CPU and GPU systems, SpMV is a primarily memory-bandwidth-bound kernel for the majority of real sparse matrices, and is bottlenecked by data movement between memory and processors [3–6].

One promising way to alleviate the data movement bottleneck is the Processing-In-Memory (PIM) paradigm [4–137]. PIM moves computation close to data by equipping memory chips with processing capabilities [8, 9, 11]. Several manufacturers have proposed *near-bank* PIM designs [4,8,138,139], that tightly couple a PIM core with each DRAM bank, exploiting bank-level parallelism to expose high on-chip memory bandwidth of standard DRAM to processors. Three *real* near-bank PIM architectures are Samsung's FIMDRAM [138], SK Hynix's GDDR6-AiM [139] and the UPMEM PIM system [4–6, 8].

Most real near-bank PIM architectures [4–6, 8, 138, 139] support several PIM-enabled memory chips connected to a host CPU via memory channels. Each memory chip comprises multiple low-power PIM cores with relatively low computation capability [4–6]. Each PIM core can access data located on its local DRAM bank, and typically there is no direct communication channel among PIM cores. Overall, near-bank PIM systems provide high levels of parallelism and very large memory bandwidth, and are thus a very promising computing platform to accelerate the widely-used SpMV kernel.

Our work is the first to efficiently map the SpMV kernel on near-bank PIM systems, and understand its performance implications on a real-world PIM system. We make two key contributions. First, we design efficient SpMV algorithms for current and future PIM systems, covering a wide variety of sparse matrices with diverse sparsity patterns. Second, we provide the first comprehensive analysis of SpMV on a real-world PIM architecture. Specifically, we conduct our rigorous analysis of SpMV kernels in the UPMEM PIM system [4, 8].

We present the openly available *SparseP* library [140] that includes 25 SpMV kernels for real PIM systems. *SparseP* supports (1) the four most popular compressed matrix formats, (2) a wide range of data types, (3) two types of well-crafted data partitioning techniques of the sparse matrix to PIM-enabled memory, (4) various load balancing schemes across PIM cores, (5) various load balancing schemes across threads of a multithreaded PIM core, and (6) three synchronization approaches among threads within multithreaded PIM core.

We conduct an extensive study of *SparseP* kernels on the UPMEM PIM system [4–6, 8]. We analyze the SpMV execution (1) using one single multithreaded PIM core, (2) using thousands of PIM cores, and (3) comparing its performance and energy consumption with that achieved on processor-centric CPU and GPU systems. Based on our rigorous experimental results and observations, we provide programming recommendations for software designers and suggestions for hardware and system designers of future PIM systems.

Our major suggestions for PIM software designers are:

1. *Design algorithms that provide high load balance across threads of a multithreaded PIM core in terms of computations, synchronization points and memory accesses.*
2. *Design compressed data structures that can be effectively partitioned across DRAM banks, with the goal of providing high computation balance across PIM cores.*
3. *Design adaptive algorithms that trade off computation balance across PIM cores for lower data transfer costs to PIM-enabled memory, and adapt the software strategies to the characteristics of both the input given and the PIM hardware.*

Our major suggestions for PIM hardware and system designers are:

1. *Provide hardware support to enable concurrent memory accesses by multiple threads to the local DRAM bank to increase parallelism in a multithreaded PIM core.*
2. *Optimize the broadcast collective operation in data transfers to PIM-enabled memory to minimize overheads of copying the input data into all DRAM banks in the PIM system.*
3. *Optimize the gather collective operation at DRAM bank granularity for data transfers from PIM-enabled memory to the host CPU to reduce overheads of retrieving the output results.*
4. *Design high-speed communication channels and optimized libraries for data transfers to/from thousands of DRAM banks of PIM-enabled memory.*

For more information about our thorough analysis on the SpMV PIM execution, insights and the open-source *SparseP* software package [140], we refer the reader to our full paper [141–143]. We hope that our work can provide valuable insights to programmers in the development of efficient sparse kernels from various application domains tailored for PIM systems, and enlighten architects and system designers in the development of future memory-centric computing systems. *SparseP* is available at https://github.com/CMU-SAFARI/SparseP.

Acknowledgments

We thank the UPMEM company for valuable support. We thank the SAFARI Research Group members for feedback and the stimulating, scholarly and collaborative intellectual environment they provide. We thank the CSLAB Research Group members for continued and undivided support, insightful comments and valuable feedback. We acknowledge the support of SAFARI Research Group's industrial partners, especially ASML, Facebook, Google, Huawei, Intel, Microsoft, VMware, the Semiconductor Research Corporation and the ETH Future Computing Laboratory. Christina Giannoula is funded for her postgraduate studies from the Foundation for Education and European Culture. This invited extended abstract is a summary version of our prior work [141] published at ACM POMACS 2022. The SparseP software package is publicly available at https://github.com/CMU-SAFARI/SparseP. A presentation that describes the work can be found at https://youtu.be/hiFkcHklLas.

References

[1] K. Kanellopoulos *et al.*, "SMASH: Co-Designing Software Compression and Hardware-Accelerated Indexing for Efficient Sparse Matrix Operations," in *MICRO*, 2019.
[2] A. Smith, "6 New Facts About Facebook," in *http://mediashift.org*, 2019.
[3] G. Goumas *et al.*, "Performance Evaluation of the Sparse Matrix-Vector Multiplication on Modern Architectures," in *J. Supercomput.*, 2009.
[4] J. Gómez-Luna *et al.*, "Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System," in *IEEE Access*, 2022.
[5] J. Gómez-Luna *et al.*, "Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-In-Memory Hardware," in *IGSC*, 2021.
[6] J. Gómez-Luna *et al.*, "Benchmarking a new paradigm: An experimental analysis of a real processing-in-memory architecture," in *CoRR*, 2021.
[7] C. Giannoula *et al.*, "SynCron: Efficient Synchronization Support for Near-Data-Processing Architectures," in *HPCA*, 2021.
[8] F. Devaux, "The True Processing In Memory Accelerator," in *Hot Chips*, 2019.
[9] O. Mutlu *et al.*, "Processing Data Where It Makes Sense: Enabling In-Memory Computation," in *MICPRO*, 2019.
[10] S. Ghose *et al.*, "Processing-in-Memory: A Workload-Driven Perspective," in *IBM JRD*, 2019.
[11] O. Mutlu *et al.*, "A Modern Primer on Processing in Memory," in *Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann*, 2021. [Online]. Available: https://arxiv.org/pdf/2012.03112.pdf
[12] H. S. Stone, "A Logic-in-Memory Computer," *IEEE TC*, 1970.
[13] W. H. Kautz, "Cellular Logic-in-Memory Arrays," *IEEE TC*, 1969.
[14] D. E. Shaw *et al.*, "The NON-VON Database Machine: A Brief Overview," *IEEE Database Eng. Bull.*, 1981.
[15] P. M. Kogge, "EXECUBE - A New Architecture for Scaleable MPPs," in *ICPP*, 1994.
[16] M. Gokhale *et al.*, "Processing in Memory: The Terasys Massively Parallel PIM Array," *IEEE Computer*, 1995.
[17] D. Patterson *et al.*, "A Case for Intelligent RAM," *IEEE Micro*, 1997.
[18] M. Oskin *et al.*, "Active Pages: A Computation Model for Intelligent Memory," in *ISCA*, 1998.
[19] Y. Kang *et al.*, "FlexRAM: Toward an Advanced Intelligent Memory System," in *ICCD*, 1999.
[20] K. Mai *et al.*, "Smart Memories: A Modular Reconfigurable Architecture," in *ISCA*, 2000.
[21] J. Draper *et al.*, "The Architecture of the DIVA Processing-in-Memory Chip," in *SC*, 2002.
[22] S. Aga *et al.*, "Compute Caches," in *HPCA*, 2017.
[23] C. Eckert *et al.*, "Neural Cache: Bit-serial In-cache Acceleration of Deep Neural Networks," in *ISCA*, 2018.
[24] D. Fujiki *et al.*, "Duality Cache for Data Parallel Acceleration," in *ISCA*, 2019.
[25] M. Kang *et al.*, "An Energy-Efficient VLSI Architecture for Pattern Recognition via Deep Embedding of Computation in SRAM," in *ICASSP*, 2014.
[26] V. Seshadri *et al.*, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," in *MICRO*, 2017.
[27] V. Seshadri *et al.*, "Buddy-RAM: Improving the Performance and Efficiency of Bulk Bitwise Operations Using DRAM," arXiv:1611.09988 [cs:AR], 2016.
[28] V. Seshadri *et al.*, "Fast Bulk Bitwise AND and OR in DRAM," *CAL*, 2015.
[29] V. Seshadri *et al.*, "RowClone: Fast and energy-efficient in-DRAM bulk data copy and initialization," in *MICRO*, 2013.
[30] S. Angizi *et al.*, "Graphide: A Graph Processing Accelerator Leveraging In-DRAM-computing," in *GLSVLSI*, 2019.
[31] J. Kim *et al.*, "The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency–Reliability Tradeoff in Modern DRAM Devices," in *HPCA*, 2018.
[32] J. Kim *et al.*, "D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput," in *HPCA*, 2019.
[33] F. Gao *et al.*, "ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs," in *MICRO*, 2019.
[34] K. K. Chang *et al.*, "Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM," in *HPCA*, 2016.
[35] X. Xin *et al.*, "ELP2IM: Efficient and Low Power Bitwise Operation Processing in DRAM," in *HPCA*, 2020.
[36] S. Li *et al.*, "DRISA: A DRAM-Based Reconfigurable In-Situ Accelerator," in *MICRO*, 2017.
[37] Q. Deng *et al.*, "DrAcc: A DRAM Based Accelerator for Accurate CNN Inference," in *DAC*, 2018.
[38] N. Hajinazar *et al.*, "SIMDRAM: A Framework for Bit-Serial SIMD Processing Using DRAM," in *ASPLOS*, 2021.
[39] S. H. S. Rezaei *et al.*, "NoM: Network-on-Memory for Inter-Bank Data Transfer in Highly-Banked Memories," in *IEEE CAL*, 2020.
[40] Y. Wang *et al.*, "FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching," in *MICRO*, 2020.
[41] M. F. Ali *et al.*, "In-Memory Low-Cost Bit-Serial Addition Using Commodity DRAM Technology," in *TCAS-I*, 2019.
[42] S. Li *et al.*, "Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-Volatile Memories," in *DAC*, 2016.
[43] S. Angizi *et al.*, "PIMA-Logic: A Novel Processing-in-Memory Architecture for Highly Flexible and Energy-efficient Logic Computation," in *DAC*, 2018.
[44] S. Angizi *et al.*, "CMP-PIM: An Energy-efficient Comparator-based Processing-in-Memory Neural Network Accelerator," in *DAC*, 2018.
[45] S. Angizi *et al.*, "AlignS: A Processing-in-Memory Accelerator for DNA Short Read Alignment Leveraging SOT-MRAM," in *DAC*, 2019.
[46] Y. Levy *et al.*, "Logic Operations in Memory Using a Memristive Akers Array," *Microelectronics Journal*, 2014.
[47] S. Kvatinsky *et al.*, "MAGIC—Memristor-Aided Logic," *IEEE TCAS II: Express Briefs*, 2014.
[48] A. Shafiee *et al.*, "ISAAC: A Convolutional Neural Network Accelerator with In-situ Analog Arithmetic in Crossbars," *ISCA*, 2016.
[49] S. Kvatinsky *et al.*, "Memristor-Based IMPLY Logic Design Procedure," in *ICCD*, 2011.
[50] S. Kvatinsky *et al.*, "Memristor-Based Material Implication (IMPLY) Logic: Design Principles and Methodologies," *TVLSI*, 2014.
[51] P.-E. Gaillardon *et al.*, "The Programmable Logic-in-Memory (PLiM) Computer," in *DATE*, 2016.
[52] D. Bhattacharjee *et al.*, "ReVAMP: ReRAM based VLIW Architecture for In-memory Computing," in *DATE*, 2017.
[53] S. Hamdioui *et al.*, "Memristor Based Computation-in-Memory Architecture for Data-intensive Applications," in *DATE*, 2015.
[54] L. Xie *et al.*, "Fast Boolean Logic Papped on Memristor Crossbar," in *ICCD*, 2015.
[55] S. Hamdioui *et al.*, "Memristor for Computing: Myth or Reality?" in *DATE*, 2017.

[56] J. Yu *et al.*, "Memristive Devices for Computation-in-Memory," in *DATE*, 2018.

[57] C. Giannoula *et al.*, "SynCron: Efficient Synchronization Support for Near-Data-Processing Architectures," in *HPCA*, 2021.

[58] I. Fernandez *et al.*, "NATSA: A Near-Data Processing Accelerator for Time Series Analysis," in *ICCD*, 2020.

[59] D. S. Cali *et al.*, "GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis," in *MICRO*, 2020.

[60] J. S. Kim *et al.*, "GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies," *BMC Genomics*, 2018.

[61] J. Ahn *et al.*, "PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture," in *ISCA*, 2015.

[62] J. Ahn *et al.*, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing," in *ISCA*, 2015.

[63] A. Boroumand *et al.*, "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks," in *ASPLOS*, 2018.

[64] A. Boroumand *et al.*, "CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators," in *ISCA*, 2019.

[65] G. Singh *et al.*, "NAPEL: Near-memory Computing Application Performance Prediction Via Ensemble Learning," in *DAC*, 2019.

[66] H. Asghari-Moghaddam *et al.*, "Chameleon: Versatile and Practical Near-DRAM Acceleration Architecture for Large Memory Systems," in *MICRO*, 2016.

[67] O. O. Babarinsa *et al.*, "JAFAR: Near-Data Processing for Databases," in *SIGMOD*, 2015.

[68] P. Chi *et al.*, "PRIME: A Novel Processing-In-Memory Architecture for Neural Network Computation In ReRAM-Based Main Memory," in *ISCA*, 2016.

[69] A. Farmahini-Farahani *et al.*, "NDA: Near-DRAM acceleration architecture leveraging commodity DRAM devices and standard memory modules," in *HPCA*, 2015.

[70] M. Gao *et al.*, "Practical Near-Data Processing for In-Memory Analytics Frameworks," in *PACT*, 2015.

[71] M. Gao *et al.*, "HRL: Efficient and Flexible Reconfigurable Logic for Near-Data Processing," in *HPCA*, 2016.

[72] B. Gu *et al.*, "Biscuit: A Framework for Near-Data Processing of Big Data Workloads," in *ISCA*, 2016.

[73] Q. Guo *et al.*, "3D-Stacked Memory-Side Acceleration: Accelerator and System Design," in *WoNDP*, 2014.

[74] M. Hashemi *et al.*, "Accelerating Dependent Cache Misses with an Enhanced Memory Controller," in *ISCA*, 2016.

[75] M. Hashemi *et al.*, "Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads," in *MICRO*, 2016.

[76] K. Hsieh *et al.*, "Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems," in *ISCA*, 2016.

[77] D. Kim *et al.*, "Neurocube: A Programmable Digital Neuromorphic Architecture with High-Density 3D Memory," in *ISCA*, 2016.

[78] G. Kim *et al.*, "Toward Standardized Near-Data Processing with Unrestricted Data Placement for GPUs," in *SC*, 2017.

[79] J. H. Lee *et al.*, "BSSync: Processing Near Memory for Machine Learning Workloads with Bounded Staleness Consistency Models," in *PACT*, 2015.

[80] Z. Liu *et al.*, "Concurrent Data Structures for Near-Memory Computing," in *SPAA*, 2017.

[81] A. Morad *et al.*, "GP-SIMD Processing-in-Memory," *ACM TACO*, 2015.

[82] L. Nai *et al.*, "GraphPIM: Enabling Instruction-Level PIM Offloading in Graph Computing Frameworks," in *HPCA*, 2017.

[83] A. Pattnaik *et al.*, "Scheduling Techniques for GPU Architectures with Processing-in-Memory Capabilities," in *PACT*, 2016.

[84] S. H. Pugsley *et al.*, "NDC: Analyzing the Impact of 3D-Stacked Memory+Logic Devices on MapReduce Workloads," in *ISPASS*, 2014.

[85] D. P. Zhang *et al.*, "TOP-PIM: Throughput-Oriented Programmable Processing in Memory," in *HPDC*, 2014.

[86] Q. Zhu *et al.*, "Accelerating Sparse Matrix-Matrix Multiplication with 3D-Stacked Logic-in-Memory Hardware," in *HPEC*, 2013.

[87] B. Akin *et al.*, "Data Reorganization in Memory Using 3D-Stacked DRAM," in *ISCA*, 2015.

[88] M. Gao *et al.*, "Tetris: Scalable and Efficient Neural Network Acceleration with 3D Memory," in *ASPLOS*, 2017.

[89] M. Drumond *et al.*, "The Mondrian Data Engine," in *ISCA*, 2017.

[90] G. Dai *et al.*, "GraphH: A Processing-in-Memory Architecture for Large-scale Graph Processing," *IEEE TCAD*, 2018.

[91] M. Zhang *et al.*, "GraphP: Reducing Communication for PIM-based Graph Processing with Efficient Data Partition," in *HPCA*, 2018.

[92] Y. Huang *et al.*, "A Heterogeneous PIM Hardware-Software Co-Design for Energy-Efficient Graph Processing," in *IPDPS*, 2020.

[93] Y. Zhuo *et al.*, "GraphQ: Scalable PIM-based Graph Processing," in *MICRO*, 2019.

[94] P. C. Santos *et al.*, "Operand Size Reconfiguration for Big Data Processing in Memory," in *DATE*, 2017.

[95] S. Ghose *et al.*, "Processing-in-Memory: A Workload-Driven Perspective," *IBM JRD*, 2019.

[96] W.-M. Hwu *et al.*, "Rebooting the Data Access Hierarchy of Computing Systems," in *ICRC*, 2017.

[97] M. Besta *et al.*, "SISA: Set-Centric Instruction Set Architecture for Graph Mining on Processing-in-Memory Systems," in *MICRO*, 2021.

[98] J. D. Ferreira *et al.*, "pLUTo: In-DRAM Lookup Tables to Enable Massively Parallel General-Purpose Computation," *arXiv:2104.07699 [cs.AR]*, 2021.

[99] A. Olgun *et al.*, "QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAMs," in *ISCA*, 2021.

[100] S. Lloyd *et al.*, "In-memory Data Rearrangement for Irregular, Data-intensive Computing," *Computer*, 2015.

[101] D. G. Elliott *et al.*, "Computational RAM: Implementing Processors in Memory," *IEEE Design & Test of Computers*, 1999.

[102] L. Zheng *et al.*, "RRAM-based TCAMs for pattern search," in *ISCAS*, 2016.

[103] J. Landgraf *et al.*, "Combining Emulation and Simulation to Evaluate a Near Memory Key/Value Lookup Accelerator," 2021.

[104] A. Rodrigues *et al.*, "Towards a Scatter-Gather Architecture: Hardware and Software Issues," in *MEMSYS*, 2019.

[105] S. Lloyd *et al.*, "Design Space Exploration of Near Memory Accelerators," in *MEMSYS*, 2018.

[106] S. Lloyd *et al.*, "Near Memory Key/Value Lookup Acceleration," in *MEMSYS*, 2017.

[107] M. Gokhale *et al.*, "Near Memory Data Structure Rearrangement," in *MEMSYS*, 2015.

[108] R. Nair *et al.*, "Active Memory Cube: A Processing-in-Memory Architecture for Exascale Systems," *IBM JRD*, 2015.

[109] A. C. Jacob *et al.*, "Compiling for the Active Memory Cube," Tech. rep. RC25644 (WAT1612-008). IBM Research Division, Tech. Rep., 2016.

[110] Z. Sura *et al.*, "Data Access Optimization in a Processing-in-Memory System," in *CF*, 2015.

[111] R. Nair, "Evolution of Memory Architecture," *Proceedings of the IEEE*, 2015.

[112] R. Balasubramonian *et al.*, "Near-Data Processing: Insights from a MICRO-46 Workshop," *IEEE Micro*, 2014.

[113] Y. Xi *et al.*, "In-Memory Learning With Analog Resistive Switching Memory: A Review and Perspective," *Proceedings of the IEEE*, 2020.

[114] K. Hsieh *et al.*, "Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation," in *ICCD*, 2016.

[115] A. Boroumand *et al.*, "LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory," *CAL*, 2016.

[116] A. Denzler *et al.*, "Casper: Accelerating stencil computation using near-cache processing," *arXiv preprint arXiv:2112.14216*, 2021.

[117] A. Boroumand *et al.*, "Polynesia: Enabling Effective Hybrid Transactional/Analytical Databases with Specialized Hardware/Software Co-Design," *arXiv:2103.00798 [cs.AR]*, 2021.

[118] A. Boroumand *et al.*, "Polynesia: Enabling Effective Hybrid Transactional Analytical Databases with Specialized Hardware Software Co-Design," in *ICDE*, 2022.

[119] G. Singh *et al.*, "FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications," *IEEE Micro*, 2021.

[120] G. Singh *et al.*, "Accelerating Weather Prediction using Near-Memory Reconfigurable Fabric," *ACM TRETS*, 2021.

[121] J. M. Herruzo *et al.*, "Enabling Fast and Energy-Efficient FM-Index Exact

Matching Using Processing-Near-Memory," *The Journal of Supercomputing*, 2021.

[122] L. Yavits *et al.*, "GIRAF: General Purpose In-Storage Resistive Associative Framework," *IEEE TPDS*, 2021.

[123] B. Asgari *et al.*, "FAFNIR: Accelerating Sparse Gathering by Using Efficient Near-Memory Intelligent Reduction," in *HPCA*, 2021.

[124] A. Boroumand *et al.*, "Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks," *arXiv preprint arXiv:2109.14320*, 2021.

[125] A. Boroumand *et al.*, "Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks," in *PACT*, 2021.

[126] A. Boroumand, "Practical Mechanisms for Reducing Processor-Memory Data Movement in Modern Workloads," Ph.D. dissertation, Carnegie Mellon University, 2020.

[127] G. Singh *et al.*, "NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling," in *FPL*, 2020.

[128] V. Seshadri *et al.*, "Simple Operations in Memory to Reduce Data Movement," in *Advances in Computers, Volume 106*, 2017.

[129] S. Diab *et al.*, "High-throughput Pairwise Alignment with the Wavefront Algorithm using Processing-in-Memory," *arXiv preprint arXiv:2204.02085*, 2022.

[130] S. Diab *et al.*, "High-throughput Pairwise Alignment with the Wavefront Algorithm using Processing-in-Memory," in *HICOMB*, 2022.

[131] D. Fujiki *et al.*, "In-Memory Data Parallel Processor," in *ASPLOS*, 2018.

[132] Y. Zha *et al.*, "Hyper-AP: Enhancing Associative Processing Through A Full-Stack Optimization," in *ISCA*, 2020.

[133] S. Ghose *et al.*, "Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions," *CoRR*, 2018.

[134] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," in *2013 5th IEEE International Memory Workshop*, 2013.

[135] O. Mutlu *et al.*, "Research Problems and Opportunities in Memory Systems," *Supercomput. Front. Innov.: Int. J.*, 2014.

[136] V. Seshadri *et al.*, "In-DRAM Bulk Bitwise Execution Engine," *CoRR*, 2019.

[137] V. Seshadri *et al.*, "Chapter Four - Simple Operations in Memory to Reduce Data Movement," ser. Advances in Computers, A. R. Hurson *et al.*, Eds. Elsevier, 2017, vol. 106, pp. 107–166.

[138] S. Lee *et al.*, "Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology: Industrial Product," in *ISCA*, 2021.

[139] S. Lee *et al.*, "A 1ynm 1.25V 8Gb, 16Gb/s/pin GDDR6-based Accelerator-in-Memory Supporting 1TFLOPS MAC Operation and Various Activation Functions for Deep-Learning Applications," in *ISSCC*, 2022.

[140] SAFARI, "*SparseP* Software Package," 2022. [Online]. Available: https://github.com/CMU-SAFARI/SparseP

[141] C. Giannoula *et al.*, "SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures," in *Proc. ACM Meas. Anal. Comput. Syst.*, 2022.

[142] C. Giannoula *et al.*, "SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Systems," in *CoRR*, 2022.

[143] C. Giannoula *et al.*, "Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Systems," in *CoRR*, 2022.