Towards an End-to-End Processing-in-DRAM Acceleration of Spectral Library Search

Tianyun Zhang, Franz Franchetti
Electrical and Computer Engineering Department Carnegie Mellon University, Pittsburgh, PA, USA
{tianyun2, franzf}@andrew.cmu.edu

***Abstract*—This work explores accelerating spectral library searches, a key mass spectrometry (MS) workload, using processing-in-memory (PIM) architectures through an end-to-end, co-designed approach. We apply signal processing techniques for pre-filtering MS data and implement a sum of absolute differences (SAD) algorithm optimized for PIM to compare spectral similarity. Our methodology is evaluated using a DRAM-based PIM simulator and compared against traditional CPU implementations. While initial results with small datasets favor CPUs, our analysis indicates potential benefits for PIM with larger, more realistic proteomics datasets. This work represents an initial step towards investigating PIM acceleration for MS applications.**

I. Introduction

Mass spectrometry (MS)-based proteomics generates vast amounts of data, with modern instruments producing millions of spectra per experiment. Traditional compute-centric architectures struggle with data movement between memory and processors during spectral library searches, a challenge that grows with increasing dataset sizes. This work explores accelerating spectral library searches using processing-in-memory (PIM) architectures through an end-to-end, co-designed approach that aims to minimize data movement and leverage parallel processing capabilities within memory.

We apply signal processing techniques for pre-filtering MS data, including discretization of mass-to-charge ratios into bins and intensity value optimization. Our implementation features a sum of absolute differences (SAD) algorithm adapted to PIM architectures to compare spectral similarity. The approach incorporates fixed-point arithmetic and reduced precision representations that align with PIM hardware constraints while maintaining acceptable mass accuracy within typical MS error tolerances. We evaluate our methodology using a DRAM-based PIM simulator.

This work represents an initial step towards investigating PIM acceleration for MS applications. Future research will focus on scaling to larger datasets, optimizing task distribution between CPU and PIM components, and further exploring low-precision techniques while maintaining acceptable accuracy for proteomics analysis. These efforts aim to establish the viability of PIM architectures for accelerating data-intensive MS workflows and broader bioinformatics applications.

This work was supported in part by the JUMP 2.0 PRISM center, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

II. Background

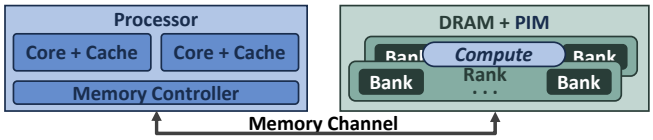

Fig. 1: Processor and DRAM+PIM architecture alleviates data movement bottleneck by adding compute units to DRAM.

DRAM and Processing-in-memory (PIM). Modern data-intensive applications across fields like bioinformatics, large-scale analytics, and machine learning increasingly process immense volumes of data. This trend has exposed limitations in traditional processor-centric architectures, where data must frequently move between physically separated processor and memory components.

In most contemporary computing systems, the processor and the memory are separated. In order to perform computations on the data stored in memory, the data must be transferred between the memory and the processor over a narrow memory channel (e.g. 64 to 128 bits wide for double data rate (DDR) DRAM) [1], [2]. Because data movement consumes orders of magnitude more energy and latency compared to on-chip computation, frequent transfers between memory and processor severely bottleneck overall system performance for workloads with large volumes of data [3].

Dynamic random-access memory (DRAM) is the most widely used main memory technology in modern computing systems because of its relatively low cost and high density. A typical DRAM-based main memory system is organized hierarchically. At the top level, a memory system connects to the processor through one or more memory channels, each controlled by a DRAM memory controller. Each channel has a dedicated, independent data, address, and command bus. A channel can host multiple Dual Inline Memory Modules (DIMMs). Within each DIMM, multiple DRAM chips are grouped to form logical units called ranks, which operate in lockstep, responding collectively to commands issued by the controller. Each device in a rank contributes a subset of bits.

Each individual DRAM chip within a rank contains multiple independently accessible data arrays called banks (16 banks in DDR4, 32 banks in DDR5). A memory bank itself refers to an independent array of DRAM cells with its own dedicated row decode logic and sense amplifiers. The presence of multiple

independent banks within a DRAM device allows concurrent operations on multiple arrays, enabling bank-level parallelism. Commands such as read, write, activate, and refresh can thus be pipelined and interleaved across different banks.

Inside each bank, memory cells are arranged in arrays comprising rows and columns. A row is a set of memory cells activated simultaneously and sensed into an array of sense amplifiers upon issuing an ACTIVATE command. Once activated, an entire row of DRAM cells is read into the associated sense amplifiers. Subsequent read and write commands then access portions of this activated row (page), avoiding repeated activation overheads. In a rank of DRAM devices operating in parallel, activating one row simultaneously activates corresponding rows across all DRAM chips in that rank. Thus, from the memory controller viewpoint, the total page size equals the row width of an individual DRAM device multiplied by how many devices exist within a rank.

Each DRAM array row further consists of multiple columns, with each column representing the smallest addressable unit of data within DRAM. During a memory access, data moves from the activated row (now held in sense amplifiers) through internal data paths onto the external bus. DDR DRAM transfers data twice per clock cycle, effectively doubling bus transfer frequency compared to single data rate SDRAM, thus increasing achievable external bandwidth. However, the narrow external data interface (e.g., typically 64 bits wide in commodity DIMMs) fundamentally constrains the maximum attainable bandwidth in conventional DDR-based memory systems [2].

PIM Taxonomy. Processing-in-memory (PIM) architectures (Fig. 1) leverage the high internal bandwidth and inherent parallelism provided by memory architectures, including DRAM and emerging memory technologies, to overcome data movement bottlenecks. PIM approaches may be broadly divided into two categories: processing-using-memory (PUM) and processing-near-memory (PNM) [1], [3].

Processing-using-memory (PUM) architectures leverage inherent circuit-level properties of memory cells to perform computation within memory arrays. Prior works have explored PUM on SRAM, DRAM, NAND flash, and emerging NVMs (e.g. RRAM, MRAM, FeRAM, PCRAM).

Processing-near-memory (PNM) architectures place computation units close to memory arrays, leveraging emerging integration technologies to enable vertically stacked DRAM layers directly connected to a logic layer using vertical interconnects called through-silicon vias (TSVs). Commercial products exemplifying PNM include Hybrid Memory Cube (HMC) and High-Bandwidth Memory (HBM), which provide greater internal bandwidth than standard DDR modules [3].

In this work, we focus on DRAM-based PIM architectures. Recent work has characterized three potential placement locations for DRAM-based PIM: subarray-level bit-serial, subarray-level bit-parallel, and bank-level.

Subarray-level bit-serial architectures place computation units at the sense amplifier level. Consequently, each bit of a row in a subarray can be processed simultaneously. Examples include DRISA [4], Micron's digital In-Memory Intelligence (IMI) [5], and DRAM-CAM [6]. These architectures often use a vertical data layout to perform SIMD-like operations at each bit position across a subarray.

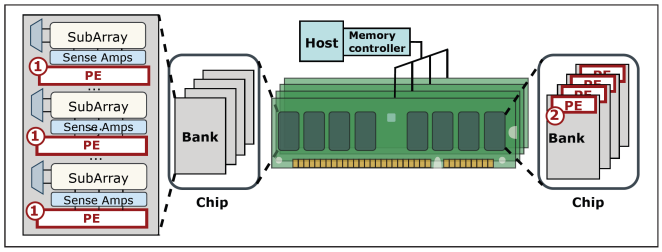

Fig. 2: Subarray-level and bank-level processing elements placements on DRAM-based PIM [1]

Subarray-level bit-parallel architectures place processing elements at the row buffer level, and are also an instance of PNM. A representative example is Fulcrum [7], which shares a 32-bit 167 MHz arithmetic logic unit (ALU) between two consecutive subarrays.

Bank-level PIM architectures place processing elements at the bank interface, and are an example of PNM. A notable example is BLIMP [8], which embeds a 200 MHz RISC-V processor at each memory bank, each capable of independently executing instructions on data stored locally within that bank's arrays.

Mass Spectrometry (MS) and Spectral Library Search. Mass spectrometry (MS) is a widely adopted method in proteomics research to identify and quantify peptides and proteins within complex biological samples [9]. Each MS measurement generates a spectrum, characterized by a set of peaks. Each peak is represented by a pair of values: the mass-to-charge ratio (m/z) and its corresponding ion intensity. The peaks represent fragment ions generated from peptides during tandem MS (MS/MS) fragmentation. Additionally, each MS/MS spectrum includes metadata, such as the precursor ion m/z, indicative of the intact peptide ion prior to fragmentation.

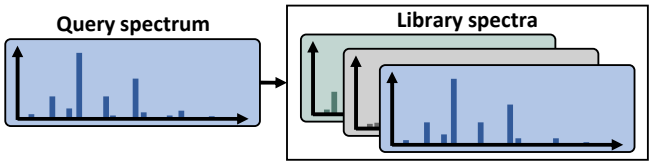

Fig. 3: Spectral library search identifies experimental spectrum by querying it against library spectra.

Spectral library search (Fig. 3) identifies experimental spectra by matching them to known reference spectra. The search process involves calculating similarity scores between an experimental spectrum and each library spectrum. A common limitation of traditional methods is their reliance on precursor mass similarity constraints, limiting identification to unmodified peptides. Open Modification Searching (OMS) improves identification by removing this constraint, thereby detecting peptides with post-translational modifications (PTMs) [9]. However, OMS dramatically enlarges the search space and

increases computational demands. With modern MS experiments generating massive data—millions of mass spectra—which must then be matched against libraries of known spectra to identify molecules, this comparison becomes expensive.

To address these performance challenges, HOMS-TC [9] introduces Hyperdimensional Computing (HDC) for spectral library search, encoding spectra into high-dimensional vectors (hypervectors) and simplifying the spectral matching process to cosine similarity search. The HDC algorithm is accelerated on tensor core units of NVIDIA GPUs.

Because the spectral library search problem involves parallel comparisons of query spectra with many library spectra, as datasets continue to scale up, it becomes a suitable candidate workload for DRAM-based PIM architectures. However, directly porting state-of-the-art OMS methods, like HOMS-TC, to PIM is challenging: these algorithms perform irregular memory access patterns and random database searches that are poorly suited to PIM architectures, which typically require regular, streaming data operations.

This work takes a fundamentally different approach, inspired by the unique strengths of bit-serial PIM architectures. Rather than relying on irregular, random access patterns, we restructure the comparison into regular, parallelizable, bit-level operations.

III. Methodology and Implementation

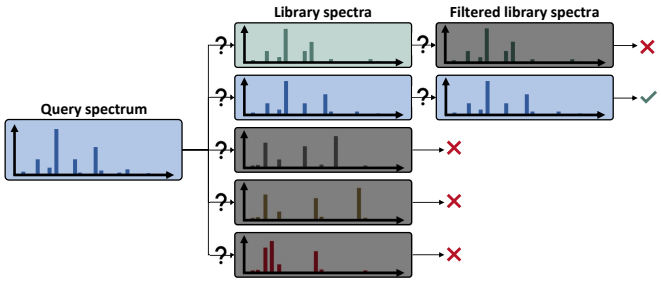

Fig. 4: Pre-filtered spectral matching decreases search space.

Pre-processing and Prefiltering. In the data pre-processing and pre-filtering step (Fig. 4), we begin by discretizing the data similarly to previous work [9], by dividing the m/z range of spectra into bins and summing intensity values of m/z values in the same bin. Specifically, we set the bin sizes as 0.05, covering an m/z range from 101 to 1500. During this binning process, low-intensity peaks, defined as having normalized intensity below 0.01, are directly filtered out to reduce noise. Up to 50 peaks per spectrum are retained (based on intensity), establishing a final maximum row length of 50 intensity-bin pairs per processed spectrum. The data can then be represented by sequences of bin index and intensity pairs.

We extend this preprocessing further by exploring approximation techniques inspired by signal processing. Given the dataset resolution, bin size, and practical experimental tolerances, we can see that employing double-precision floating-point arithmetic is excessively precise for such experimental tasks.

Specifically, typical instrumentation used in our experiments tolerates precursor mass errors around 2 ppm for exact search, and up to 50 ppm for open modification search [10], [11]. At typical protein-centric MS mass ranges, a ppm-level tolerance means that a precision of less than 6 decimal digits of accuracy is more than sufficient. Therefore, standard double-precision floating-point representations, which provide around 15 decimal digit precision, considerably exceed our actual experimental accuracy demands.

Therefore, we experiment with reducing the bit-precision of floating point values and using fixed-point representation for those values. Fixed-point representation simplifies the hardware necessary to process the data and allows for the application to be run on PIM hardware.

Minimum Sum of Absolute Differences (SAD). We use an adapted minimum SAD algorithm optimized for execution on PIM architectures. Variations of this algorithm are commonly used in the signal processing domain to measure similarity between images. We use SAD to calculate a similarity score between the experimental spectrum $S = (s_1, s_2, \ldots, s_n)$ with each library spectrum $L_i = (\ell_{i1}, \ell_{i2}, \ldots, \ell_{in})$ for $i = 1, 2, \ldots, m$. Assuming m library spectra and n elements in each spectrum, the basic algorithm seeks L_k such that:

$$k = \arg\min_i \sum_{j=1}^{n} |s_j - \ell_{ij}|$$

We implement minimum SAD using PIM API instructions from PIMeval, a performance and energy simulator for diverse PIM architectures [1]. We model the algorithm using a bit-serial PIM approach, because the vertical data layout used by bit-serial architectures facilitates flexible bit-precision operations. Each bit of a multi-bit number is stored vertically along a column within a DRAM subarray, allowing multiple operations to execute simultaneously across bit-slices. To increase the algorithm's noise tolerance, we also enable SAD with shifted indices for inexact matching, implemented using the rightward rotation (`pimRotateElementsRight`) instruction:

```
for (int idx=0; idx < subvecLen; idx++) {
  pimSub(obj1, obj2, obj3);
  pimAbs(obj3, obj3);
  for (int i=idx; i+subvecLen-1 < vecLen; i+=subvecLen) {
    pimRedSumRanged(obj3, i, i+subvecLen-1, &sumAbsDiff);
    ...
  }
  pimRotateElementsRight(obj2);
}
```

IV. Evaluation

Architectural Simulation. We modeled subarray-level bit-serial PIM for the application on the simulator and conducted initial experiments comparing our PIM-based approach with a traditional CPU implementation for SAD, using our pre-processed data. The parameters used for the PIM device were a single rank DIMM with 8 chips, 16 banks per chip, and 32 subarrays per bank of 8192×8192 cells.

For the current small-scale problem of 11 MB, the CPU implementation completed execution in 175 ms, while the PIM implementation took 1368 ms for the core computation tasks.

Our experiments with a 2-rank DIMM configuration for the same problem size revealed that, at this scale, the workload underutilized the available parallelism of the additional hardware resources. These preliminary results highlight an important aspect of PIM architectures: their performance benefits are expected to scale with problem size. While the current small dataset favors the CPU, we anticipate that for larger, more realistic proteomics problem sizes, PIM acceleration will demonstrate advantages in scalability, memory bandwidth utilization, and energy efficiency, being able to benefit from parallel processing and decreases in data movement. Ongoing work will quantify these projected improvements by simulating much larger datasets reflective of real-world proteomics experiments.

Algorithm Robustness and Discussion of ROC Results. To thoroughly evaluate the robustness of our minimum SAD-based approach, we designed an experiment to evaluate robustness under degradation that simulates spectral variations. We conducted experiments where we systematically varied parameters, introducing random peak removal, intensity noise, and mass-to-charge (m/z) bin jitter. Our goal was to assess to what extent the SAD metric can discriminate the correct, original version of a spectrum from a set of other unrelated spectra under these challenging conditions.

The results, shown in the receiver operating characteristic (ROC) curve in Figure 5, demonstrate reliable performance, with Area Under the Curve (AUC) greater than 0.96 under moderate noise.

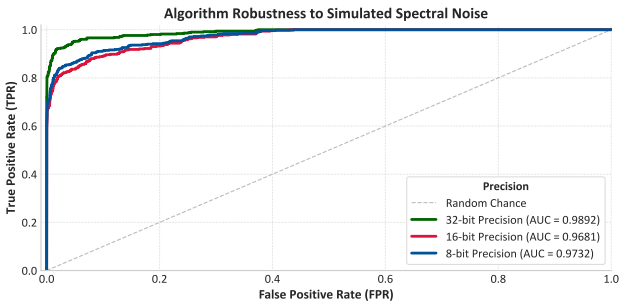

Fig. 5: ROC curves demonstrating the robustness of minimum SAD variant algorithm under moderate simulated noise. The plot shows the trade-off between identifying true matches (TPR) and incorrectly identifying false matches (FPR) at different precisions.

To test whether this pattern holds outside moderate-noise, we repeated the experiment across a wider range of degradation levels. The resulting AUCs are summarized in Table I. As expected, the "No-Noise" control shows that all precisions perform perfectly (AUC=1.0) in the absence of noise. Under High-Noise conditions, the AUC decreases to the upper 0.7 range. These results indicate that 8-bit can deliver comparable accuracy to higher precisions in the scenarios tested, though the impact of precision may grow for more rigorous conditions and different datasets, and merits further study.

TABLE I: Algorithm Robustness under Varying Degradation

Experiment Condition	**Degradation** (Rem/Jit/Noise)	**AUC (32-bit)**	**AUC (16-bit)**	**AUC (8-bit)**
No-Noise Control	0% / ±0 / 0%	**1.0000**	1.0000	1.0000
Moderate Noise	10% / ±1 / 10%	**0.9892**	0.9681	0.9732
High-Noise Test	25% / ±2 / 25%	**0.7819**	0.7583	0.7606

Discussion on Scaling. Finally, we note the limitations in the scope of our current simulation-based experiments. Due to restrictions in the hardware simulation tools, our evaluations have only examined limited problem sizes and architectural configurations. Furthermore, the effective problem size for our analysis is significantly impacted by the preprocessing pipeline. The standard practice of retaining only the top 50 peaks results in a data reduction of up to 91.9%, which while helpful for analysis, means that the final datasets do not yet represent the challenge of large-scale data movement that PIM aims to overcome. The next step is to estimate, through analytical calculations or extended simulator experiments, the data set size and the required DRAM capacity at which clear performance gains from the bit-serial PIM architecture will become evident.

Ultimately, the performance benefits of our proposed bit-serial PIM algorithm are expected to scale and become significant primarily in large-scale deployments with broader parallelism and much larger datasets. The current evaluations represent initial steps toward validating the feasibility, correctness, and robustness of our algorithm rather than offering a complete, large-scale performance analysis. Nevertheless, these preliminary validations confirm robust behavior and correctness, laying the groundwork for subsequent large-scale evaluations.

Conclusion and Future Work

We have demonstrated a proof of concept for an end-to-end, hardware-algorithm co-designed pipeline for DRAM-based PIM architecture acceleration of spectral library search, a key workload for mass spectrometry. The next step is to observe the performance from taking better advantage of the massive parallelism in DRAM for more reasonable comparisons and to connect larger volumes of data from real-world workloads to the hardware for a full-system view of PIM accelerated bioinformatics applications.

References

[1] F. A. Siddique, D. Guo, Z. Fan, M. Gholamrezaei, M. Baradaran, A. Ahmed, H. Abbot, K. Durrer, K. Nandagopal, E. Ermovick *et al.*, "Architectural modeling and benchmarking for digital dram pim."

[2] B. Jacob, D. Wang, and S. Ng, *Memory systems: cache, DRAM, disk*. Morgan Kaufmann, 2010.

[3] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "A modern primer on processing in memory," in *Emerging Computing: From Devices to Systems*. Springer, 2023, pp. 171–243.

[4] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "Drisa: A dram-based reconfigurable in-situ accelerator," in *Proceedings of the 50th annual ieee/acm international symposium on microarchitecture*, 2017, pp. 288–301.

[5] T. Finkbeiner, G. Hush, T. Larsen, P. Lea, J. Leidel, and T. Manning, "In-memory intelligence," *IEEE micro*, vol. 37, no. 4, pp. 30–38, 2017.

[6] L. Wu, R. Sharifi, A. Venkat, and K. Skadron, "Dram-cam: General-purpose bit-serial exact pattern matching," *IEEE Computer Architecture Letters*, vol. 21, no. 2, pp. 89–92, 2022.

[7] M. Lenjani, P. Gonzalez, E. Sadredini, S. Li, Y. Xie, A. Akel, S. Eilert, M. R. Stan, and K. Skadron, "Fulcrum: A simplified control and access mechanism toward flexible and practical in-situ accelerators," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 556–569.

[8] A. Devic, S. B. Rai, A. Sivasubramaniam, A. Akel, S. Eilert, and J. Eno, "To pim or not for emerging general purpose processing in ddr memory systems," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 2022, pp. 231–244.

[9] J. Kang, W. Xu, W. Bittremieux, N. Moshiri, and T. Rosing, "Accelerating open modification spectral library searching on tensor core in high-dimensional space," *Bioinformatics*, vol. 39, no. 7, p. btad404, 2023.

[10] V. R. Pagala, A. A. High, X. Wang, H. Tan, K. Kodali, A. Mishra, K. Kavdia, Y. Xu, Z. Wu, and J. Peng, "Quantitative protein analysis by mass spectrometry," *Protein-protein interactions: Methods and applications*, pp. 281–305, 2015.

[11] J. M. Chick, D. Kolippakkam, D. P. Nusinow, B. Zhai, R. Rad, E. L. Huttlin, and S. P. Gygi, "A mass-tolerant database search identifies a large proportion of unassigned spectra in shotgun proteomics as modified peptides," *Nature biotechnology*, vol. 33, no. 7, pp. 743–749, 2015.