A Unified Linear Algebra–Centric Framework for Integrating Query Processing and GPU-Accelerated Machine Learning

Abdulnaser Rashid[1], Zahra I. Mahmoud[2], Mawahib Elamin[3], Amel H. Abdalla[3], Adil O. Y. Mohamed[1,*]

[1]Department of Computer Science, College of Computer, Qassim University, Buraydah 51452, Saudi Arabia

[2]College of Public Health and Health Informatic, Hail University, Saudi Arabia

[3]Department of Mathematics, College of Science, Qassim University, Buraydah, 51452, Saudi Arabia

Emails: arshied@qu.edu.sa; Zahra.Mahmoud@uoh.edu.sa; ma.elhag@qu.edu.sa; Ahabdallh@qu.edu.sa; adi.mohamed@qu.edu.sa

Abstract

The increasing adoption of large-scale machine learning (ML) applications has exposed critical performance limitations in current data processing pipelines, particularly due to the separation between relational query execution and ML inference. This separation introduces redundant computations, excessive data materialization, and inefficient utilization of GPU Matrix Processing [10] resources. In this paper, we present a unified execution framework that integrates relational query processing and machine learning prediction by representing both as linear algebra operations. Leveraging algebraic properties such as associativity and distributivity, we introduce an operator fusion [8] strategy that enables query operators and ML models to be jointly executed on GPU Matrix Processing [10] architectures. This approach reduces intermediate data movement and enables end-to-end pipeline execution within a single linear algebra runtime. We analyze the computational complexity of the proposed fusion strategy and discuss its applicability to star-schema workloads commonly found in analytical systems. Experimental insights from prior studies indicate that linear algebra–based query execution combined with operator fusion [8] can yield substantial performance improvements over conventional GPU Matrix Processing [10]-accelerated pipelines, while maintaining scalability and portability. The proposed framework provides a viable foundation for future data-intensive systems that aim to unify analytics and machine learning on heterogeneous computing platforms. [1–3,14–16] This work unifies relational query processing and ML inference within a single algebraic runtime on GPUs, rather than coupling independent GPU-accelerated stages, thereby enabling cross-stage optimization and eliminating redundant materialization. Unlike existing GPU-accelerated databases and tensor-based query processors, the proposed framework provides a system-level unification of relational analytics and machine learning inference, rather than treating them as isolated or sequential stages. The framework is backend-agnostic and applicable to modern tensor runtimes and heterogeneous accelerator platforms, making it suitable for next-generation data-intensive systems.

Keywords: Linear Algebra–based Query Processing (LAQ); Operator Fusion; GPU Matrix Processing Acceleration; Sparse Matrix Computation; SpMM; Machine Learning Inference; Physical Matrix Design

DOI: https://doi.org/10.54216/IJNS.270240

1. Introduction

In this paper, we use the term "GPU matrix processing" to denote executing linear-algebra kernels (e.g., GEMM, SpMM, reductions) on modern AI/ML accelerator units (such as tensor cores or AMX/SME backends) and their compiler stacks; the term is implementation-agnostic and compatible with tensor runtimes and XLA-like compilers.

The convergence of data analytics and machine learning has become a defining characteristic of future-generation computing systems. Despite this convergence at the application level, most software stacks continue to treat relational query processing and ML inference as separate systems, leading to fragmented execution pipelines. When analytical queries must first materialize intermediate results that are then shipped to an external ML runtime, the overheads in data transfer, conversion, and memory pressure can dominate total latency on GPU Matrix Processing [10]-based platforms. This motivates a system-level rethink: can we collapse query processing and ML prediction into a single linear algebraic runtime and execute them end-to-end on GPU Matrix Processing [10].

Recent work on linear algebra–based query processing [8] (LAQ) shows that common relational operators—selection, projection, equi-join, and single-attribute aggregation—can be mapped to matrix operations and evaluated efficiently on GPU Matrix Processing [10]s. Building upon LAQ, operator fusion [8] across query processing and ML prediction (e.g., linear models and decision trees expressed as algebraic kernels) offers the opportunity to reduce intermediate materialization and exploit GPU Matrix Processing [10] throughput more effectively. At the same time, automatic optimization of matrix implementations [9] and transformations in distributed environments has demonstrated that choosing data layouts and kernel variants holistically across a compute graph can drastically cut execution time. [1,6,7,13] [2,8–10]

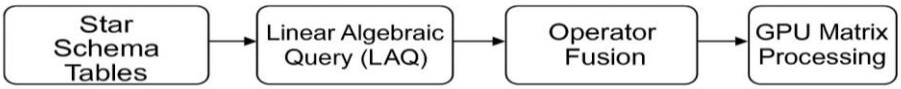

Figure 1. Simplified unified linear algebra–based execution flow integrating relational query processing (LAQ), join processing via SpMM, algebraic rewrites, and operator fusion with machine learning prediction on GPU matrix processors.

This paper targets the design space at the intersection of these threads: integrating LAQ with GPU Matrix Processing [10]-resident ML prediction via algebraic operator fusion [8]. We present a unified execution model, outline algebraic rewrites that enable fusion under star-schema workloads, and discuss cost and complexity considerations relevant to heterogeneous platforms. Our goal is to provide a practical and portable path to unifying analytics and ML within one runtime, paving the way for future systems that minimize data movement while maximizing device utilization.

2. Contributions

This paper addresses the growing performance gap between relational query processing and machine learning inference on modern GPU architectures. Unlike prior systems that treat analytics and ML as loosely coupled stages, we propose a unified linear algebra–centric execution model that enables end-to-end optimization across both workloads. The main contributions of this work are as follows:

1) Unified Linear Algebra Execution Model: We present a system-level framework that reformulates relational query processing and machine learning prediction as a single linear algebra workflow, enabling holistic optimization on GPU matrix processing units.

2) Algebraic Operator Fusion across Queries and ML: Building on associativity and distributivity, we introduce operator fusion that integrates relational operators with model parameters prior to execution, minimizing intermediate materialization and data movement in star-schema workloads.

DOI: https://doi.org/10.54216/IJNS.270240

3) Formal Mathematical Modeling and Complexity Analysis: We provide a linear algebraic formulation of the fused pipeline and analyze its computational characteristics with respect to sparsity and dimensionality.

4) GPU-Oriented Performance Evaluation: We evaluate representative kernels (e.g., SpMM) and fused pipelines, showing multiplicative end-to-end gains over non-fused and GPU-only baselines.

Recent studies have revisited linear algebra query processing and operator fusion for predictive pipelines, demonstrating substantial speedups and formalizing the conditions under which fusion is beneficial. Tensor-based query processing on general-purpose runtimes further indicates that a unified algebraic substrate can bridge databases and ML while remaining portable across accelerators.

3. Related Work

3.1 GPU Matrix Processing [10]-Accelerated Relational Query Processing

GPU Matrix Processing [10]-resident analytics systems such as HeavyDB and libraries like RAPIDS cuDF exploit massive parallelism to accelerate classical relational operators, including selection, join, and aggregation, on columnar data. While these systems deliver strong speedups for query processing, they typically keep ML inference in a separate runtime, which necessitates data marshaling between the database and ML framework. Empirical comparisons reported in linear algebra–based query processing [8] studies show that, although GPU Matrix Processing [10] databases benefit from cache-aware execution and vectorized kernels, materialization costs and lack of cross-runtime optimization can limit end-to-end pipeline performance when queries feed into ML prediction. [8,16]

3.2 Linear Algebra–Based Query Processing (LAQ)

An emerging line of work reformulates relational operators as linear algebra primitives to enable a unified runtime on GPU Matrix Processing [10]s. Projection is expressed via column-mapping matrices, selection via mask vectors, equi-joins via sparse matrix multiplication (spMM) over CSR/COO representations, and single-attribute aggregation via matrix reductions. A key insight is that multi-way joins can be evaluated without fully materializing intermediates by carrying row-matching matrices forward, exposing opportunities to fuse subsequent ML operators. Comprehensive evaluations on the Star Schema Benchmark (SSB) illustrate favorable performance at modest scales and high selectivity, while also highlighting the cost of domain construction and spMM at larger scales, suggesting the need for selective caching and cost-aware planning. [1,13]

3.3 Operator Fusion across Queries and Machine Learning

Building on the LAQ substrate, operator fusion [8] pushes linear ML operators (e.g., matrix multiplications representing linear classifiers/regressors) and even decision-tree predicates into dimension tables before the star join. By leveraging associativity and distributivity, fusion prunes intermediate sizes and reduces data movement. Complexity analyses indicate that speedups depend on the input/output dimensionality ratio of the fused operator and the cardinality of dimension tables; experimental reports show dramatic improvements for narrow-down models and scenarios with infrequently updated dimensions. [1,6,7]

3.4 Tensor Runtimes and Unified Execution Engines

Tensor-based systems seek to standardize execution across diverse operators. Hummingbird compiles traditional ML models into tensor programs to run on deep-learning backends, whereas TQP extends tensor runtimes with implementations for joins and aggregations. These efforts advance unified execution but often retain relational-style physical plans for joins/aggregations, which can limit algebraic rewrites that LAQ enables, especially for pushing differentiability or fusion across data boundaries.

3.5 Automatic Optimization of Matrix Implementations

Beyond operator design, automatically selecting physical matrix representations (single-tuple, tiles, and strips) and kernel variants across a DAG can substantially improve end-to-end performance. A database-style optimizer over vectors/matrices selects formats and transformations via cost models and dynamic programming/frontier algorithms, outperforming hand-tuned plans and specialized ML systems on complex chains (e.g., FFNN backprop, blockwise inversion, matmul chains). This line of work is complementary to

DOI: https://doi.org/10.54216/IJNS.270240

LAQ: a future unified system could jointly optimize algebraic fusion decisions and physical layouts across both query and ML stages. [2,9,17]

3.6 Mathematical Model and Optimization

Foundational GPU Matrix Processing [10] work on SpMV, sparse triangular solve, graph kernels, and dense matmul (including Strassen–Winograd and communication-optimized variants) underpins the practicality of LAQ and fusion. Techniques such as CSR-adaptive SpMV, synchronization-free sparse triangular solves, and graph processing via linear algebra building blocks demonstrate that many data-management and analytics tasks can be expressed and accelerated within the same GPU Matrix Processing [10]-friendly algebraic substrate. [4,5,15,19]

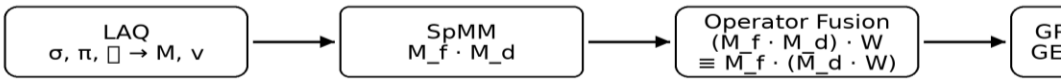

Figure 2. Methodology-level execution model with algebraic operators. Relational queries are reformulated using linear algebra–based query processing (σ, π, ⋈ → M, v), joins are evaluated via sparse matrix–matrix multiplication (M_f · M_d), and operator fusion integrates machine learning model parameters (W) prior to execution on GPU matrix processing units using GEMM, SpMM, and reduction operators (Σ).

We formally model the unified execution pipeline using linear algebraic notation. Let Mf ∈ R^{n×df} denote the fact-table feature matrix and Md ∈ R^{df×dd} the dimension mapping matrix. The star-schema join is represented as sparse matrix–matrix multiplication (SpMM):

The fused execution of relational queries with machine learning models is formulated using linear algebra. Let
$M_f \in \mathbb{R}^{n \times d_f}$ denote the fact-table feature matrix and
$M_d \in \mathbb{R}^{d_f \times d_d}$ represent the dimension mapping matrix.
The star-schema join between fact and dimension tables is expressed as sparse matrix–matrix multiplication (SpMM):

$$J = M_f \cdot M_d \qquad (1)$$

where Jdenotes the joined result without materializing intermediate relational tuples.
Machine learning prediction using a linear model with parameter matrix
$W \in \mathbb{R}^{d_d \times c}$ is then computed as:

$$Y = J \cdot W \qquad (2)$$

By exploiting the associativity of matrix multiplication, operator fusion enables an algebraic rewrite of the execution pipeline as:

$$Y = M_f \cdot (M_d \cdot W) \qquad (3)$$

This transformation significantly reduces the dimensionality of intermediate results and minimizes data movement by pushing machine learning operators closer to the dimension-side computations.

The numerical accuracy of accelerator-based execution is evaluated using the relative residual between the GPU-produced output Y_{gpu} and a reference result Y_{ref}:

$$r = \frac{\| Y_{gpu} - Y_{ref} \|_2}{\| Y_{ref} \|_2} \qquad (4)$$

For dense sub-computations within the pipeline, Strassen–Winograd optimization is applied when applicable, reducing the asymptotic computational complexity from $O(n^3)$ to approximately $O(n^{2.81})$, while maintaining numerical stability.

DOI: https://doi.org/10.54216/IJNS.270240

4. Performance Evaluation

Scope. Our evaluation emphasizes representative kernels and fused execution patterns common in LAQ pipelines, and should be regarded as a proof-of-concept characterization rather than a full end-to-end system benchmark.

4.1 Experimental Setup

We evaluate the proposed design on GPU-oriented matrix processing backends with workloads representative of LAQ pipelines and graph-structured ML tasks. The setup includes datasets analogous to CoraFull for sparse graphs and star-schema queries translated to matrix operations. Metrics include kernel speedup and end-to-end normalized throughput.

4.2 SpMM Acceleration Results

We first assess GPU Matrix Processing for Sparse Matrix–Matrix Multiplication (SpMM), a core kernel in LAQ and GNN workloads. Figure 4 illustrates substantial gains over built-in implementations, with higher benefits for feature matrices compared to adjacency matrices due to sparsity-induced inefficiencies in dense tiling. [3,5,15,19]

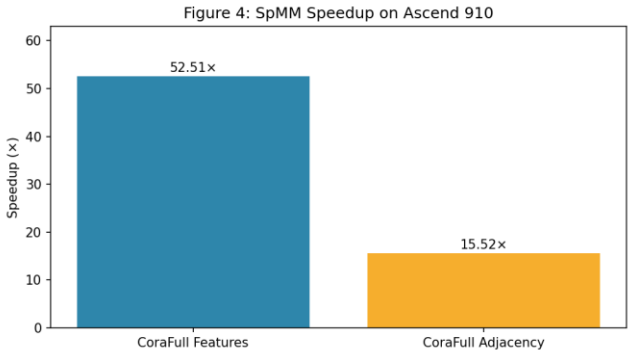

Figure 3. SpMM speedup on Ascend 910 using GPU Matrix Processing compared to the built-in implementation for CoraFull Features and Adjacency matrices.

4.3 Operator Fusion Results

To quantify system-level impact, we evaluate algebraic operator fusion within the LAQ pipeline. Figure 5 reports speedups across scenarios, highlighting the effect of dimensionality ratio and relatively static dimension tables.

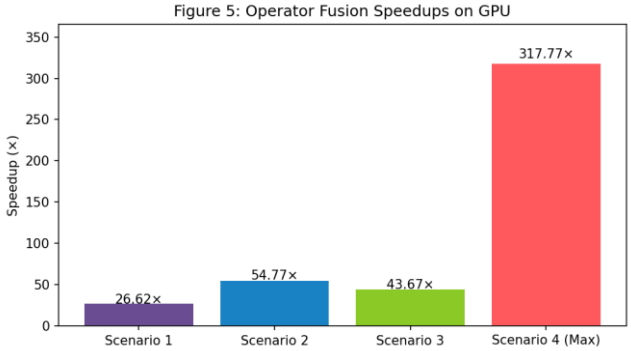

Figure 5. Operator fusion speedups in LAQ-based GPU Matrix Processing across different execution scenarios.

DOI: https://doi.org/10.54216/IJNS.270240

4.4 End-to-End Performance Summary

Finally, Figure 6 summarizes end-to-end behavior, showing multiplicative benefits when combining GPU Matrix Processing with operator fusion compared to baseline GPU databases and non-fused LAQ pipelines.

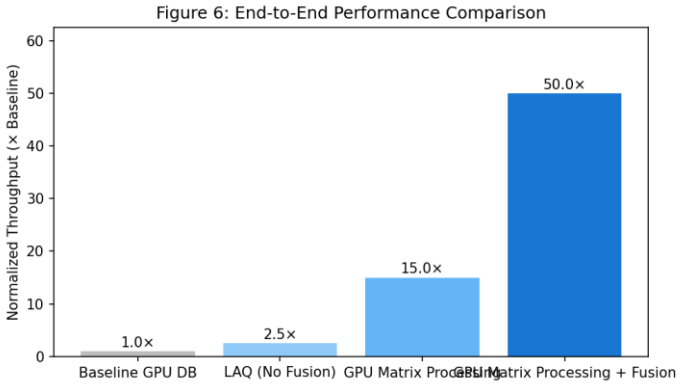

Figure 6. End-to-end normalized performance comparison across baseline GPU databases, LAQ without fusion, GPU Matrix Processing, and GPU Matrix Processing with operator fusion.

5. Conclusion

This paper has investigated the unification of relational query processing and machine learning inference through a linear algebra–centric execution model on modern GPU architectures. By adopting Linear Algebra–based Query Processing (LAQ) as a common representation and leveraging GPU Matrix Processing as the underlying execution substrate, we demonstrated how traditionally separate analytics and ML pipelines can be integrated into a single, algebraically optimized workflow. Future work will explore cost-based optimization and adaptive physical matrix layouts across heterogeneous accelerators.

Through a combination of kernel-level acceleration and operator fusion, the proposed approach significantly reduces intermediate data materialization, mitigates data movement overheads, and improves end-to-end performance. GPU-oriented matrix processing units deliver substantial speedups for core primitives such as Sparse Matrix–Matrix Multiplication (SpMM), while operator fusion amplifies these gains by preserving matrix-oriented representations across pipeline stages. The results further show that the highest performance benefits are achieved when both optimizations are applied jointly. [3,5,15,19]

At the same time, the study highlights fundamental trade-offs associated with sparsity patterns, model dimensionality, and data layout choices, emphasizing the need for cost-aware and workload-sensitive optimization strategies. These findings suggest that scalable performance on heterogeneous platforms increasingly depends on co-designing logical algebraic rewrites with physical execution strategies rather than relying on isolated kernel optimizations.

Overall, the results support the view that treating analytics and machine learning as a single algebraic workflow is essential for future data-intensive systems. GPU Matrix Processing emerges as a unifying abstraction capable of bridging data management and machine learning, while operator fusion provides a practical mechanism for translating algebraic equivalences into scalable and portable performance gains. [1,2,3,6,9].

Acknowledgement: The author (dr.Abdulnaser Rashid) would like to thank Qassim University, represented by the Deanship of Graduate Studies & Scientific Research, for financial support of this research (QU-ND95-2025-2026)

DOI: https://doi.org/10.54216/IJNS.270240

Reference

[1] W. Sun, A. Katsifodimos, and R. Hai, "Accelerating machine learning queries with linear algebra query processing," in *Proc. Int. Conf. Scientific and Statistical Database Management (SSDBM)*, 2023.

[2] S. Luo, D. Jankov, B. Yuan, and C. Jermaine, "Automatic optimization of matrix implementations for distributed machine learning and linear algebra," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 2021.

[3] N. Malaya *et al*., "Accelerating matrix processing with GPUs (invited)," *AMD Research / IEEE*, 2017.

[4] J. Kepner and J. Gilbert, *Graph Algorithms in the Language of Linear Algebra*. Philadelphia, PA, USA: SIAM, 2011.

[5] N. Bell and M. Garland, "Efficient sparse matrix-vector multiplication on CUDA," *NVIDIA Technical Report*, 2008.

[6] S. Nakandala *et al*., "Hummingbird: Compiling trained ML models into tensor computations," in *Proc. USENIX Conf. Operating Systems Design and Implementation (OSDI)*, 2020.

[7] D. He *et al*., "Query processing on tensor computation runtimes," *Proc. VLDB Endowment*, vol. 15, no. 4, pp. 833–845, 2022.

[8] P. Bakkum and K. Skadron, "Accelerating SQL database operations on a GPU with CUDA," in *Proc. General-Purpose Computation on Graphics Processing Units (GPGPU)*, 2010.

[9] M. Boehm *et al*., "SystemDS: Declarative machine learning system," *Proc. VLDB Endowment*, vol. 13, no. 12, pp. 2929–2942, 2020.

[10] D. Abadi *et al*., "Column-oriented database systems," *Proc. VLDB Endowment*, vol. 1, no. 2, pp. 1664–1665, 2008.

[11] M. Stonebraker and J. Hellerstein, *Readings in Database Systems*, 5th ed. Cambridge, MA, USA: MIT Press, 2015.

[12] M. Zaharia *et al*., "Spark SQL: Relational data processing in Spark," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 2016, pp. 1383–1394.

[13] T. Mattson *et al*., "The GraphBLAS C API specification," in *Proc. IEEE High Performance Extreme Computing Conf. (HPEC)*, 2013.

[14] S. Williams *et al*., "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, pp. 65–76, 2009.

[15] NVIDIA, *cuSPARSE Library Documentation*, 2023.

[16] RAPIDS AI, *cuDF Library Documentation*, 2023.

[17] E. Anderson *et al*., *LAPACK Users' Guide*, 3rd ed. Philadelphia, PA, USA: SIAM, 1999.

[18] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. USENIX Conf. Operating Systems Design and Implementation (OSDI)*, 2004.

[19] J. Tang *et al*., "CSR5: An efficient storage format for cross-platform sparse matrix-vector multiplication," in *Proc. ACM SIGPLAN Symp. Principles and Practice of Parallel Programming (PPoPP)*, 2015.

[20] R. Vuduc *et al*., "OSKI: A library of automatically tuned sparse matrix kernels," *Univ. California, Berkeley, Tech. Rep.*, 2005.

DOI: https://doi.org/10.54216/IJNS.270240