Relational Processing of Tensor Programs

Thomas Muñoz Serrano [1]

[1] Data Science Institute, Hasselt University, Belgium
{thomas.munozserrano}@uhasselt.be

Diverse data science applications [1, 2, 3, 4, 5, 6, 7, 8] require the ability to operate over numerical arrays that have more than two dimensions, commonly known as *tensors*. Motivated by this necessity, in recent years there has been an emergence of tensor compilers such as TACO [9], Finch [18], Halide [11] and ExTensor [12], as well as declarative frameworks like Galley [13].

In addition, real-world numerical data is sparse: tensors with predominantly zero entries. Similarly, relational data is stored sparsely: only relevant tuples are materialized. For this reason, relational query engines arise as a natural candidate to process tensor workloads, since they are inherently sparse-oriented, performing computation only over existing tuples.

The systems mentioned above extend array programming (e.g., NumPy [14] and PyTorch [15]) to sparse data, supporting arbitrary user-defined pointwise functions and aggregates. With the exception of Galley, these compilers focus on systematically optimizing the computation tensor kernels, not arbitrary tensor expressions. Tensor kernels are common tensor expressions that arise in different contexts and work as building blocks for tensor-based algorithms. Current systems [9, 18, 11, 12, 13] do not attempt to adapt relational processing optimization techniques to the computation of the tensor kernels.

On the other hand, many of the solutions that have tackled numerical computations from a relational perspective focus on processing linear algebra computations, and assume that matrix entries and operations belong to an arbitrary semiring [9, 16, 17, 18, 19, 20, 21, 22, 27, 23, 25, 26, 24, 16, 28], and system-oriented linear algebra optimizations [27, 23, 25, 26, 24, 16, 28] assume that entries and functions belong to the usual, but specific, semiring $(\mathbb{R}, \cdot, +)$. If we want to go beyond a single semiring, the FAQ [29] framework is an option, but the expressions are restricted to one pointwise funtction, which is insufficient to express even the most basic tensor kernels compositions.

To the best of our knowledge, there is no formal characterization of tractable fragments when using multiple pointwise functions on tensor programs. At first, by tractability we mean computable by a relational algebra program. For example, given a tensor program that mixes ReLU, min, and arithmetic operations, under what conditions can we guarantee that its evaluation and updates can be expressed and optimized within relational techniques if we are only interested in computing tensor values that lie within a certain range (sparse computation)?

Our main result provides a general characterization of the functions for which a full tensor query can be reduced to a relational algebra computation. Specifically, we show that every such query can be translated into an expression using *union*, *join*, and *set difference*. Moreover, within this class we identify a fragment of functions for which set difference is not required, so that the reduction involves only *union* and *join*. These characterizations establish the first bridge between tensor queries and relational query evaluation.

References

[1] Blackford, L. S., Petitet, A., Pozo, R., Remington, K., Whaley, R. C., Demmel, J., Dongarra, J., Duff, I., Hammarling, S., Henry, G., & others (2002). An updated set of basic linear algebra subprograms (BLAS). *ACM Transactions on Mathematical Software*, 28(2), 135–151.

[2] Mattson, T., Bader, D., Berry, J., Buluc, A., Dongarra, J., Faloutsos, C., Feo, J., Gilbert, J., Gonzalez, J., Hendrickson, B., & others (2013). Standards for graph algorithm primitives. In *2013 IEEE High Performance Extreme Computing Conference (HPEC)* (pp. 1–2). IEEE.

[3] Azad, A., Buluç, A., & Gilbert, J. (2015). Parallel triangle counting and enumeration using matrix algebra. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop* (pp. 804–811). IEEE.

[4] Zhao, H. (2014). *High performance machine learning through codesign and rooflining* [Doctoral dissertation, University of California, Berkeley].

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

[6] Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3), 455–500.

[7] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[8] Smith, S., Ravindran, N., Sidiropoulos, N. D., & Karypis, G. (2015). SPLATT: Efficient and parallel sparse tensor-matrix multiplication. In *2015 IEEE International Parallel and Distributed Processing Symposium* (pp. 61–70). IEEE.

[9] Kjolstad, F., Kamil, S., Chou, S., Lugato, D., & Amarasinghe, S. (2017). The Tensor Algebra Compiler. *Proceedings of the ACM on Programming Languages (OOPSLA)*, 1(OOPSLA), 77:1–77:29.

[10] Ahrens, W., Collin, T. F., Patel, R., Deeds, K., Hong, C., & Amarasinghe, S. (2024). Finch: Sparse and structured array programming with control flow. arXiv preprint arXiv:2404.16730.

[11] Ragan-Kelley, J., Barnes, C., Adams, A., Paris, S., Durand, F., & Amarasinghe, S. (2013). Halide: A Language and Compiler for Optimizing Parallelism, Locality, and Recomputation. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)* (pp. 519–530).

[12] Hegde, K., Asghari-Moghaddam, H., Pellauer, M., Crago, N., Jaleel, A., Solomonik, E., Emer, J., & Fletcher, C. W. (2019). ExTensor: An Accelerator for Sparse Tensor Algebra. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture* (pp. 319–333). ACM.

[13] Deeds, K., Ahrens, W., Balazinska, M., & Suciu, D. (2025). Galley: Modern Query Optimization for Sparse Tensor Programs. *Proceedings of the ACM on Management of Data*, 3(3), 164, 1–24.

[14] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., & others (2020). Array Programming with NumPy. *Nature*, 585(7825), 357–362.

[15] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 32.

[16] Chen, L., Kumar, A., Naughton, J., & Patel, J. M. (2017). Towards Linear Algebra over Normalized Data. arXiv preprint arXiv:1612.07448.

[17] Mattson, T., Davis, T. A., Kumar, M., Buluc, A., McMillan, S., Moreira, J., & Yang, C. (2019). LAGraph: A Community Effort to Collect Graph Algorithms Built on Top of the GraphBLAS. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (pp. 276–284).

[18] Ahrens, W., Collin, T. F., Patel, R., Deeds, K., Hong, C., & Amarasinghe, S. (2024). Finch: Sparse and structured array programming with control flow. arXiv preprint arXiv:2404.16730.

[19] Buluç, A., Mattson, T., McMillan, S., Moreira, J., & Yang, C. (2017). Design of the GraphBLAS API for C. In *2017 IEEE international parallel and distributed processing symposium workshops (IPDPSW)* (pp. 643–652). IEEE.

[20] Green, T. J., Karvounarakis, G., & Tannen, V. (2007). Provenance semirings. In L. Libkin (Ed.), *Proceedings of the 26th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS 2007, Beijing, China, June 11-13, 2007* (pp. 31–40). ACM.

[21] Brijder, R., Geerts, F., Van den Bussche, J., & Weerwag, T. (2019). On the Expressive Power of Query Languages for Matrices. *ACM Transactions on Database Systems*, 44(4), 15:1–15:31.

[22] Geerts, F., Muñoz, T., Riveros, C., & Vrgoc, D. (2021). Expressive Power of Linear Algebra Query Languages. In L. Libkin, R. Pichler, & P. Guagliardo (Eds.), *Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2021, Virtual Event, China, June 20-25, 2021* (pp. 342–354). ACM.

[23] Kanellopoulos, K., Vijaykumar, N., Giannoula, C., Azizi, R., Koppula, S., Mansouri-Ghiasi, N., Shahroodi, T., Gómez-Luna, J., & Mutlu, O. (2019). SMASH: Co-designing Software Compression and Hardware-Accelerated Indexing for Efficient Sparse Matrix Operations. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2019, Columbus, OH, USA, October 12-16, 2019* (pp. 600–614). ACM.

[24] Luo, S., Gao, Z. J., Gubanov, M. N., Perez, L. L., & Jermaine, C. M. (2018). Scalable Linear Algebra on a Relational Database System. *SIGMOD Record*, 47(1), 24–31.

[25] Wang, Y. R., Hutchison, S., Suciu, D., Howe, B., & Leang, J. (2020). SPORES: Sum-Product Optimization via Relational Equality Saturation for Large Scale Linear Algebra. *Proceedings of the VLDB Endowment*, 13(11), 1919–1932.

[26] Jankov, D., Luo, S., Yuan, B., Cai, Z., Zou, J., Jermaine, C., & Gao, Z. J. (2020). Declarative Recursive Computation on an RDBMS: or, Why You Should Use a Database For Distributed Machine Learning. *SIGMOD Record*, 49(1), 43–50.

[27] Huang, B., Babu, S., & Yang, J. (2013). Cumulon: optimizing statistical data analysis in the cloud. In K. A. Ross, D. Srivastava, & D. Papadias (Eds.), *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013* (pp. 1–12). ACM.

[28] Boehm, M., Kumar, A., & Yang, J. (2019). *Data Management in Machine Learning Systems*. Morgan & Claypool Publishers.

[29] Abo Khamis, M., Ngo, H. Q., & Rudra, A. (2016). FAQ: Questions Asked Frequently. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)* (pp. 13–28).