Mapping Unstructured Sparse Computation to Dataflow Architectures

Maya Taylor and Luke Olson
Department of Computer Science
University of Illinois at Urbana-Champaign
mayat4@illinois.edu, lukeo@illinois.edu

I. Introduction

Sparse linear algebra kernels are foundational to scientific computing, from linear solvers to graph algorithms. Despite their importance, these kernels are notoriously difficult to accelerate, particularly due to the irregular memory access patterns that drive them. This memory-boundedness has persisted across CPU and GPU generations and implementations, as shared memory management and cache hierarchies struggle with irregular access.

Spatial dataflow architectures offer a new perspective: by eliminating the cache hierarchy, giving each processing element (PE) exclusive local memory, and providing an extremely low-latency on-chip network, they re-frame sparse computation as a data placement problem — one that has the potential to reward careful algorithm design with substantial performance gains. While numerical methods for structured sparsity on dataflow have seen substantial study — including my own prior work — my current focus is on the largely unexplored frontier of unstructured sparse computation on dataflow, where the mapping challenges are harder and the potential gains less well understood. Specifically, my initial focus is on unstructured sparse matrix-vector products (SpMV) and sparse triangular solves (SpTRSV), two kernels that appear consistently throughout scientific computing workloads.

II. Background

Dataflow architectures — featuring a 2D mesh of lightweight PEs, distributed on-chip SRAM, and low-latency inter-PE communication — offer an alternative to conventional memory hierarchies for scientific computing. Examples relevant to our work include the Cerebras Wafer-Scale Engine and Tenstorrent Wormhole [1]. Unlike cache-based architectures, these platforms expose data movement explicitly, making locality a primary concern of algorithm design. Here, locality refers to the alignment between an algorithm's data access patterns and the physical layout of PEs on the mesh. Algorithms exploiting this architecture-aware locality can largely reduce the cost of communication and achieve unprecedented throughput. This has been demonstrated with the neural network workloads for which these accelerators were intended [2], but recent work has also shown strong performance gains for more scientific workloads.

The most striking numerical results have come from workloads with regular, structured data access patterns. Stencil computations, for example, map naturally to the 2D PE mesh: each PE owns a contiguous subdomain, and communication is limited to nearest-neighbor exchanges along the mesh. This locality-preserving mapping has enabled near-perfect weak scaling for stencil workloads on both the WSE and Wormhole [3]–[5]. Similar success has been demonstrated for a variety of other numerical kernels that take advantage of locality in their algorithm design [6]–[9].

My own work contributes to this body of structured results with an implementation of a conjugate gradient solver on Tenstorrent's Wormhole architecture [10]. This work characterizes the programming challenges inherent to the Wormhole architecture and outlines algorithmic choices that preserve locality and scaling properties. Critically, it also exposes the limits of the structured approach: generalizing beyond stencils to arbitrary sparse operators requires confronting problems that structured algorithms sidestep entirely. This observation motivates the central question that guides my current work: *how can unstructured sparse operations be efficiently mapped to dataflow architectures, and can the performance gains demonstrated for structured sparsity be recovered?*

III. The Mapping Problem

Mapping unstructured sparse kernels to dataflow architectures presents various interesting challenges.:

- The **irregular communication patterns** inherent to unstructured sparsity make data layout and routing on the 2D mesh a non-trivial design problem — unlike stencils, there is no natural mapping that preserves locality for all operators. This challenge lives at the heart of this work.
- **Load balance** raises a question unique to the dataflow setting: when does nonzero imbalance across PEs matter, and when is it acceptable — or even desirable — to leave lightweight PEs idle rather than redistribute work?
- Some problems introduce **data-dependent control flow**, wherein the order in which PEs can make progress depends on the sparsity structure of the problem, making static optimizations difficult.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Department of Energy Computational Science Graduate Fellowship under Award Number DE-SC0025528.

A. Methodology and Evaluation

We focus primarily on the Cerebras WSE in this work. Our approach involves decomposing various unstructured sparse operations into 1D collectives, which are simple to implement and model on the WSE [11]. Specifically, we consider 1D reductions and 1D broadcasts across a row or column of PEs, and develop algorithms around these primitives. We consider a range of different numerical algorithms that map to these 1D collectives with varying ease. Of our two initial case studies, SpMV lies on the simpler end of this spectrum while SpTRSV introduces dependency complexities that push towards the other end.

To evaluate these various algorithms, we propose using corresponding structured algorithms as a baseline and measuring the performance of our unstructured algorithms on *structured* problems in comparison. This approach aims to capture the cost of unstructured algorithms in comparison to those that make assumptions of sparsity structure. We aim to (1) quantify any overhead and additionally (2) account for it appropriately. This evaluation will likely also make use of traditional scaling and performance model methodologies.

IV. Case Study 1: SpMV

Unstructured SpMV is a natural starting point for this work because it maps cleanly to 1D broadcast and reduction on the PE grid. Given a sparse matrix A and dense vector x, our algorithm employs a column and row locality-preserving (CRLP) data layout for A, where all non-zeros in a matrix row reside on a single row of PEs and all non-zeros from a given matrix column reside on a single column of PEs. The entries of x are distributed along the top row of the grid such that they align with the corresponding columns of A.

Given this layout, the SpMV can be computed using first a series of column-wise broadcasts, which migrate each x_j to the PEs owning the corresponding non-zeros A_{ij}. Every PE then computes the relevant partial products $A_{ij}x_j$, and contributes to a row-wise 1D reduction accumulating $(Ax)_i$.

The simplicity of this mapping makes SpMV a useful tool to explore different aspects of unstructured sparsity on the WSE. As an example, consider the potential for 'sparsity-aware' 1D collectives: during a given reduction or broadcast, various PEs may not contribute or require data and can therefore be avoided all together to reduce latency. To support this, PEs on the WSE can be configured in a low-latency 'pass-over' mode, effectively reducing them to network links. Our ongoing work explores different approaches to integrate this pass-over mode into the 1D collectives and optimize them corresponding to the sparsity patterns of the matrix.

V. Case Study 2: SpTRSV

Sparse triangular solve (SpTRSV) is a fundamental kernel in scientific computing and has been extensively studied in many parallel and distributed contexts [12]–[17]. We propose an algorithm for SpTRSV on the WSE to solve the lower triangular system $Lx = b$ wherein L is laid out in a CRLP fashion similar to above. Under this layout, forward substitution through L reduces to a cycle of two 1D collective operations: a column-wise broadcast to propagate solution values x_i to new rows of L, and a row-wise reduction to accumulate partial contributions $\sum_j L_{ij}x_j$. After every row reduction is complete, a new solution value x_i is computed and redistributed and the cycle repeats.

A. Level-set-aware Layouts

Exploiting level-set structure is the primary mechanism for extracting parallelism from the otherwise sequential forward substitution in SpTRSV. We explore mappings of level sets to the WSE PE grid, looking for layouts that consider both data locality and parallel utilization.

In the simplest mapping, consecutive level sets are assigned to consecutive PE rows such that solution values propagate down the grid as the solve progresses. Since parallelism in SpTRSV is restricted to within a level-set, this layout restricts parallelism across PE rows, but allows pipe-lined solves within PE rows.

In evaluation, row pipelining has shown meaningful performance gains, correlating strongly with the number of matrix rows per level set, as expected. While this approach limits parallel utilization, it may be particularly well-suited to applications requiring many successive SpTRSV solves, such as iterative refinement or repeated preconditioner application, where multiple solves can be pipelined across the grid.

Alternative level-set mappings are under active exploration. In particular, distributing rows of a level-set across distinct PE rows instead exposes substantially more parallelism, at the cost of more data movement. Understanding these tradeoffs — and identifying which mappings best exploit the WSE's low-latency network for different sparsity structures — is an ongoing focus of this work.

VI. Current Status and Future Work

At present, we have working implementations of SpMV and SpTRSV and are in the process of exploring the optimizations discussed above. Current evaluation is focused on benchmarking these algorithms at small scale on the cycle-accurate WSE simulator across matrices from the SuiteSparse collection, and validating early results with simple performance modeling [18]. A third case study for future work is potentially asynchronous iterative methods on the WSE [19]. The fine-grained, naturally asynchronous nature of dataflow architectures may be a surprisingly effective platform for such methods, and such a case study will further test the utility of our 1D collective methodology.

As a whole, the case studies presented here represent early steps toward characterizing when and how unstructured sparse computation can be efficiently mapped to dataflow architectures. Our methodology aims to test the limits of the 1D primitives and potentially prompt new approaches for algorithm design on dataflow.

REFERENCES

[1] G. Lauterbach, "The path to successful wafer-scale integration: The cerebras story," *IEEE Micro*, vol. 41, no. 6, pp. 52–57, 2021.

[2] Z. Zhang, D. Parikh, Y. Zhang, and V. Prasanna, "Benchmarking the performance of large language models on the cerebras wafer scale engine," 2024. [Online]. Available: https://arxiv.org/abs/2409.00287

[3] N. Brown and R. Barton, "Accelerating stencils on the tenstorrent grayskull risc-v accelerator," 2024. [Online]. Available: https://arxiv.org/abs/2409.18835

[4] J. de Fine Licht, A. Kuster, T. D. Matteis, T. Ben-Nun, D. Hofer, and T. Hoefler, "Stencilflow: Mapping large stencil programs to distributed spatial computing systems," 2021. [Online]. Available: https://arxiv.org/abs/2010.15218

[5] K. Rocki, D. V. Essendelft, I. Sharapov, R. Schreiber, M. Morrison, V. Kibardin, A. Portnoy, J. F. Dietiker, M. Syamlal, and M. James, "Fast stencil-code computation on a wafer-scale processor," 2020. [Online]. Available: https://arxiv.org/abs/2010.03660

[6] R. Sai, M. Jacquelin, F. P. Hamon, M. Araya-Polo, and R. R. Settgast, "Massively distributed finite-volume flux computation," 2023. [Online]. Available: https://arxiv.org/abs/2304.11274

[7] M. Orenes-Vera, I. Sharapov, R. Schreiber, M. Jacquelin, P. Vandermersch, and S. Chetlur, "Wafer-scale fast fourier transforms," in *Proceedings of the 37th ACM International Conference on Supercomputing*, ser. ICS '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 180–191. [Online]. Available: https://doi.org/10.1145/3577193.3593708

[8] R. Sai, F. P. Hamon, J. Mellor-Crummey, and M. Araya-Polo, "Matrix-free finite volume kernels on a dataflow architecture," 2024. [Online]. Available: https://arxiv.org/abs/2408.03452

[9] K. Santos, S. Moore, T. Oppelstrup, A. Sharifian, I. Sharapov, A. Thompson, D. Z. Kalchev, D. Perez, R. Schreiber, S. Pakin, E. A. Leon, J. H. Laros, M. James, and S. Rajamanickam, "Breaking the molecular dynamics timescale barrier using a wafer-scale system," in *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, Nov. 2024, p. 1–13. [Online]. Available: http://dx.doi.org/10.1109/SC41406.2024.00014

[10] M. Taylor, C. Pearson, L. Berger-Vergiat, G. Long, and J. Ciesko, "Numerical kernels on a spatial accelerator: A study of tenstorrent wormhole," 2026. [Online]. Available: https://arxiv.org/abs/2603.23343

[11] P. Luczynski, L. Gianinazzi, P. Iff, L. Wilson, D. De Sensi, and T. Hoefler, "Near-optimal wafer-scale reduce," in *Proceedings of the 33rd International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '24. ACM, 2024, p. 334–347. [Online]. Available: http://dx.doi.org/10.1145/3625549.3658693

[12] Z. Lu, Y. Niu, and W. Liu, "Efficient Block Algorithms for Parallel Sparse Triangular Solve," in *49th International Conference on Parallel Processing - ICPP*. Edmonton AB Canada: ACM, Aug. 2020, pp. 1–11. [Online]. Available: https://dl.acm.org/doi/10.1145/3404397.3404413

[13] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu, "A Supernodal Approach to Sparse Partial Pivoting," *SIAM Journal on Matrix Analysis and Applications*, vol. 20, no. 3, pp. 720–755, Jan. 1999. [Online]. Available: http://epubs.siam.org/doi/10.1137/S0895479895291765

[14] T. Böhnlein, P. A. Papp, R. S. Steiner, C. K. Matzoros, and A. N. Yzelman, "Efficient Parallel Scheduling for Sparse Triangular Solvers," Jun. 2025, arXiv:2503.05408 [cs]. [Online]. Available: http://arxiv.org/abs/2503.05408

[15] A. Picciau, G. E. Inggs, J. Wickerson, E. C. Kerrigan, and G. A. Constantinides, "Balancing Locality and Concurrency: Solving Sparse Triangular Systems on GPUs," in *2016 IEEE 23rd International Conference on High Performance Computing (HiPC)*. Hyderabad, India: IEEE, Dec. 2016, pp. 183–192. [Online]. Available: http://ieeexplore.ieee.org/document/7839683/

[16] X. S. Li and J. W. Demmel, "Superlu_dist: A scalable distributed-memory sparse direct solver for unsymmetric linear systems," *ACM Trans. Math. Softw.*, vol. 29, no. 2, p. 110–140, Jun. 2003. [Online]. Available: https://doi.org/10.1145/779359.779361

[17] J. Mayer, "Parallel algorithms for solving linear systems with sparse triangular matrices," *Computing*, vol. 86, no. 4, pp. 291–312, Nov. 2009. [Online]. Available: http://link.springer.com/10.1007/s00607-009-0066-3

[18] T. A. Davis, "SuiteSparse: a suite of sparse matrix software," https://github.com/DrTimothyAldenDavis/SuiteSparse, 2026.

[19] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1989.