Mapping Unstructured Sparse Computation to Dataflow Architectures

Maya Taylor, Luke Olson
Department of Computer Science, University of Illinois at Urbana-Champaign

IPDPS 2026 • New Orleans, USA

DOE CSGF

ILLINOIS

Abstract

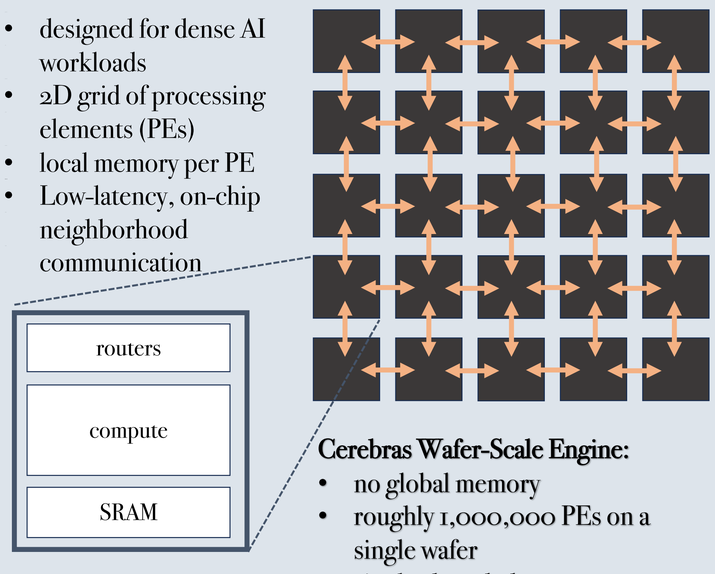
Sparse linear algebra kernels are foundational to scientific computing but notoriously difficult to accelerate, particularly due to the **irregular memory access patterns** that drive them.

Spatial **dataflow architectures** offer a new perspective. They re-frame sparse computation as a data placement problem – one that has the potential to reward careful algorithm design with substantial performance gains.

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.

Dataflow Architectures

- designed for dense AI workloads
- 2D grid of processing elements (PEs)
- local memory per PE
- Low-latency, on-chip neighborhood communication



Cerebras Wafer-Scale Engine:
- no global memory
- roughly 1,000,000 PEs on a single wafer
- single-threaded

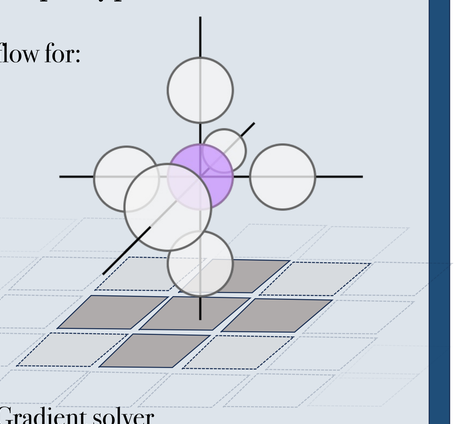
Tenstorrent Wormhole:
- PE-local SRAM
- limited global DRAM
- roughly 100 PEs per Wormhole grid
- multi-core PEs

Programming models:
- many dataflow architectures don't support traditional languages/compilers
- asynchronous, data driven programming

Structured Sparsity on Dataflow

Despite being designed for AI, dataflow architectures have proven to be a good fit for numerical workloads, including **sparse linear algebra** with **structured, regular sparsity patterns**.

Existing work explores Dataflow for:
- **stencil-like workloads,**
- sparse linear solvers,
- finite volume methods,
- Fast Fourier Transform,
- molecular dynamics



My work contributes to these structured results on the Tenstorrent Wormhole:
- stencil-based Conjugate Gradient solver
- 7-point stencil operator
- simple Jacobi preconditioner

These workloads take advantage of **known static sparsity patterns** to implement sparsity-aware operators that are super efficient on Dataflow.

Unstructured Sparsity

Many important sparse problems don't exhibit these regular and predictable sparsity pattern.

These problems are difficult to map to Dataflow because they exhibit:
1. irregular communication patterns
2. load imbalance
3. data-dependent control flow

The approach to each of these challenges may be specific to a specific iteration of a Dataflow architecture.

Research Questions

- How do we develop algorithms for unstructured sparse computation on Dataflow?
- Are there certain types of unstructured sparsity that Dataflow is better suited for?
- Can we recover the same exciting performance exhibited for structured operators?

Methodology

- 1D collectives over rows and columns of the grid offer a good starting point for our algorithm design for Dataflow
- we explore how to map a variety of unstructured sparse problems to the **Cerebras WSE** using 1D collectives
- the chosen problems vary in how simply they map to the 1D collectives
- eventually this algorithmic exploration may use to other approaches

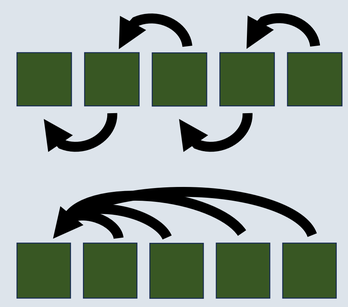
Evaluation

- traditional scaling and performance modeling
- performance comparisons with traditional architectures (both runtime and other aspects, such as power consumption)
- evaluating the proposed **unstructured algorithms on structured problems,** and comparing them to structure-aware implementations

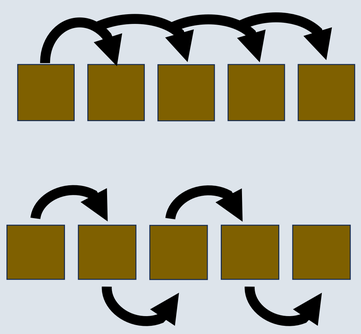
1D Collectives

- simple building blocks
- easy to **implement** and **model**
- variations to explore
- potential for sparsity-aware alternatives

Chain Reduction:
- easy to pipeline
- distributed computation
- not sparsity-aware



Star Reduction:
- bottleneck at root
- no pipelining
- can be sparsity-aware
- uses the fast pass-over routing mode



Multicast Broadcast:
- fast multicast routing mode
- only one predetermined PE can inject
- can be made sparsity-aware

Chain Broadcast:
- slower pass-through routing
- any PE can inject into the broadcast
- not sparsity-aware

Sparse Matrix-Vector Product

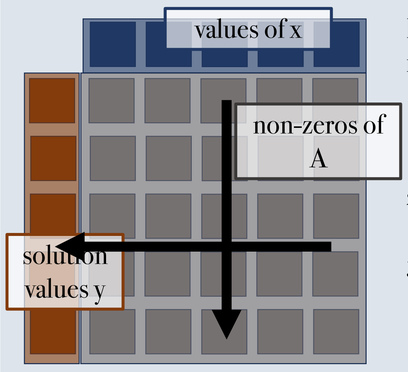
SpMV is a good starting point because it maps well to the proposed 1D collectives. Specifically, consider the product:

$$Ax = y$$

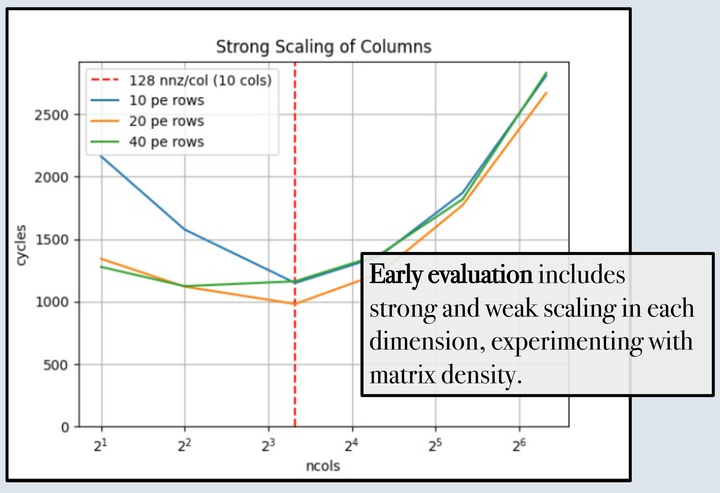
where A: unstructured and sparse, x: dense.

Row and Column-Locality Preserving Mappings:
Map the matrix A to a 2D mesh such that:
- non-zeros in the same matrix row reside on PEs in the same mesh row
- non-zeros in the same matrix column reside on PEs in the same mesh column

Vector x is distributed across a PE row, aligning with columns of A.



Dataflow algorithm:
1. x values are broadcast down grid columns and matched with corresponding non-zeros of A
2. PEs compute local partial products
3. solution values y are computed via row-wise reductions



Early evaluation includes strong and weak scaling in each dimension, experimenting with matrix density.

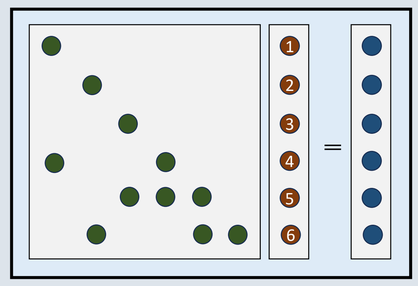
Sparse Triangular Solve

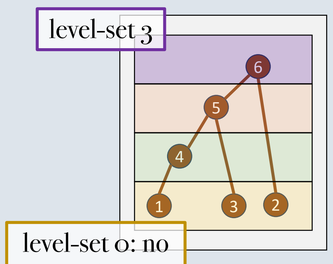
SpTRSV introduces added complexity in the dependency chain imposed by sparsity structure. Consider the solve for x:

$$Lx = b$$

where L is unstructured, sparse, and lower triangular, and b is dense.

Structure of L determines DAG of dependencies and level-sets.





level-set 0: no dependencies

Mapping:
- similarly to SpMV, use a row and column-locality preserving mapping to map L to the main grid of PEs
- load balance wrt L's triangular structure, for example by using a cyclic row and column distribution
- order rows by level-set such that low level-sets are physically placed above higher-level sets
- level-set 0 is isolated as it requires no communication

Algorithm:
1. solution values are broadcast through the appropriate column
2. PEs compute local partial products for forward-substitution
3. new solution values are computed via row-wise reductions
4. solution values are then broadcast back to the appropriate column
5. the process repeats

Future Work

- currently, we have implemented SpMV and SpTRSV and begun evaluation on the WSE simulator
- we are actively exploring various optimizations including sparsity-aware 1D collectives and different level-set mappings
- we plan to evaluate our unstructured algorithms on…
  - much larger examples on true WSE hardware
  - equivalent structured problems, building a comprehensive breakdown of performance loss
- later work includes exploration of algorithm design outside of the 1D primitive structure

SCAN ME