Sandia National Laboratories

SPARSE LINEAR SOLVERS ON TENSTORRENT'S DATAFLOW ARCHITECTURE

Maya Taylor[1], Carl Pearson[2], Luc Berger-Vergiat[2], Jan Ciesko[2]
[1]University of Illinois at Urbana-Champaign, [2]Sandia National Labs

We implement **Richardson iteration** on Tenstorrent's **Wormhole.** Performance results show scaling capability and comparisons to more traditional architectures.

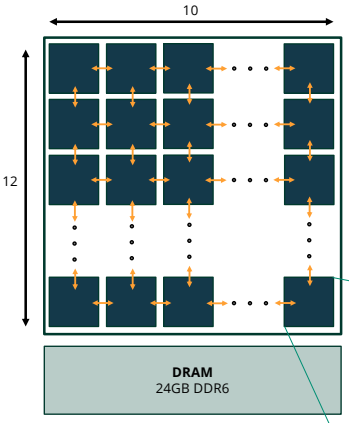
Motivation

- Designed for AI, dataflow architectures show promise when repurposed for scientific computing.
- Richardson iteration is a good proxy for the data movement and compute patterns of general iterative solvers.

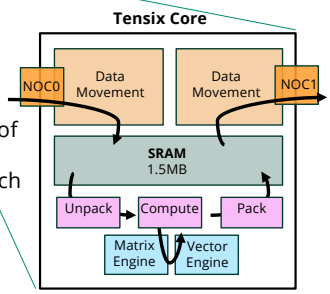
What are the **challenges** posed by the *hardware* and *programming interface*?

What are the **benefits** of a *spatial* architecture for sparse linear algebra?

Architecture



- The Wormhole n300 is a PCIe board with **80** compute-designated cores, and global **DRAM.**
- Every Tensix core is connected to its 4 neighbors in a **2D Torus NOC.**
- Data is viewed as 32x32 tiles of 16-bit Bfloats by default. Up to 32-bit FP is supported.



- Every Tensix core is made up of 5 RISC-V cores, SRAM, and a matrix and vector engine, which operate on 2D tiled registers.
- Each RISC-V core is single threaded and serves a unique control flow purpose.
- Data movement cores: bring data in and out of the Tensix
- Unpack: moves data from SRAM into compute registers
- Compute: initiates matrix/vector computation
- Pack: moves data from registers to SRAM

Richardson Iteration

- A fixed-point iterative method to solve the linear system:

$$Ax = b$$

- Richardson iteration iteratively updates the solution x until convergence:

$$x_0 = x(0)$$
$$x_{k+1} = x_k + \omega(b - Ax_k)$$

- We solve **the 1D Laplacian**, so that A is **tridiagonal**, with 2 on the diagonal and -1 on the off diagonals.
- Proxy for the communication and computation patters of more popular iterative solvers, like **Conjugate Gradient**

Implementation

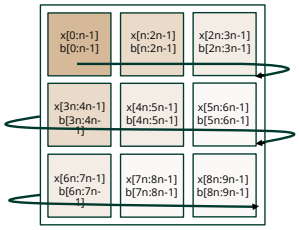
- The initial **x** and **b** vectors are mapped to Tensix cores in rows
- Data is divided into **tiles** of 1024 elements, *n* per core.
- Tenstorrent's low-level programming language **tt-metal** provides tile-wise operations

Richardson iteration requires:

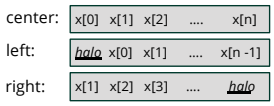
Element-wise operations
- add, subtract, multiply
- provided directly by tt-metal

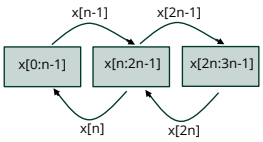
Norm
- for checking convergence
- *reduction*: collected at top left, aggregated at every core along the way, to reduce data movement
- *broadcast*: provided by tt-metal

SPMV
- implemented for the tridiagonal structure of the given A
- Create 2 copies of a given tile, one shifted left and one shifted right
- Add these three tiles with the correct scaling factors (-1 and 2)
- At every iteration, halo data is retrieved from left and right neighbors via core-to-core comm



$$Ax[i] = -x[i-1] + 2x[i] - x[i+1]$$

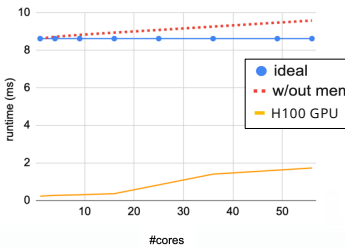




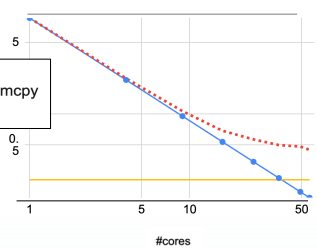
Performance

- On a single Wormhole chip, we scale to a 7x7 sub grid of Tensix.
- Limited by the L1 size, each Tensix has capacity for 165 tiles (165 * 1024 elements) of input data.
- Performance shows ideal weak and strong scaling for 20 iters.
- Bottleneck: the process of **shifting tiles (memcpy)** left and right for SPMV, limited by the core's 16B memory alignment requirement. We are working on various strategies to eliminate this shifted tile construction cost and leave it off scaling plots.

Weak Scaling



Strong Scaling



Conclusion

- Wormhole's grid-like architecture shows promise for efficient, fast stencil-based algorithms.
- Future work includes solving memory access issues and other optimizations, a power study, and a 27-point stencil Conjugate Gradient.

Acknowledgements
This work was supported by *Ramesh Ganisetti* and the Tenstorrent team, *Giovanni Long* and his work on profiler integration, and *Clay Hughes* and *Alex Council* at Sandia.