SpMV for AMG on the Cerebras Wafer-Scale Engine

Maya Taylor, Luke Olson

University of Illinois at Urbana-Champaign

AMG + Cerebras

- Lots of ways to do AMG on CPUs and GPUs!
- We're interested in a new accelerator: the Cerebras Wafer-Scale Engine (WSE)

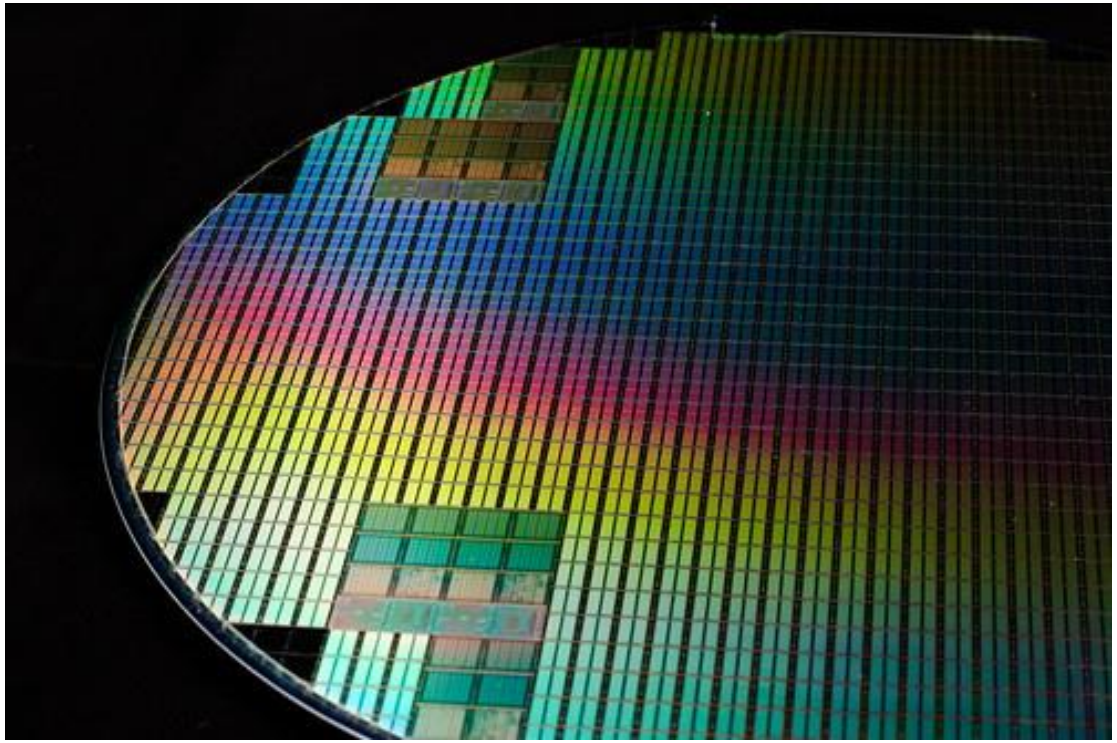
- Explore the potential for AMG on the WSE
  - through SpMV
  - what happens on Cerebras at coarse grid levels

In this talk…

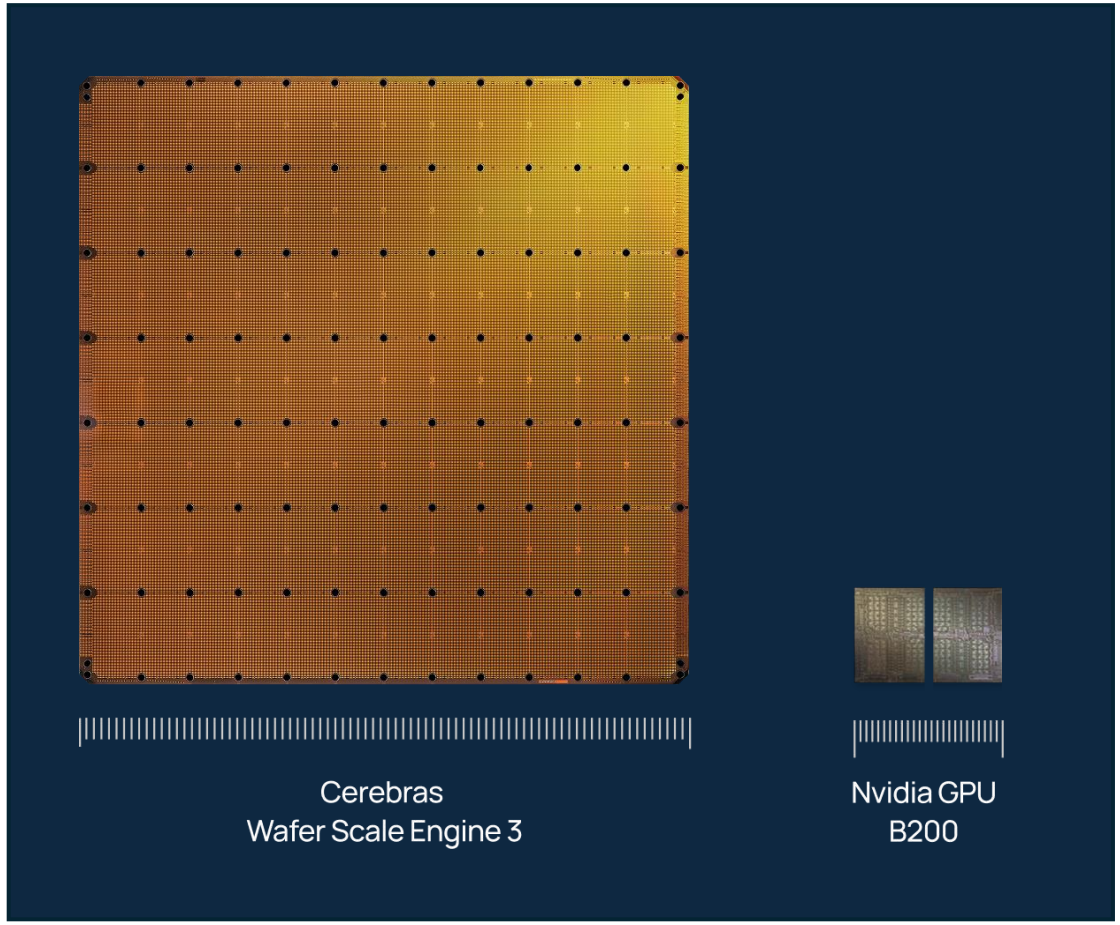
- an overview of the Cerebras WSE architecture
- our proposed algorithm for general SpMV on the WSE
- early evaluation of the WSE SpMV algorithm
- algorithmic considerations for the AMG context

‘Wafer-Scale’?

- Typical manufacturing process cuts a wafer into many small components
- Wafer-scale:
  - utilize maximal wafer area
  - keep communication on silicon



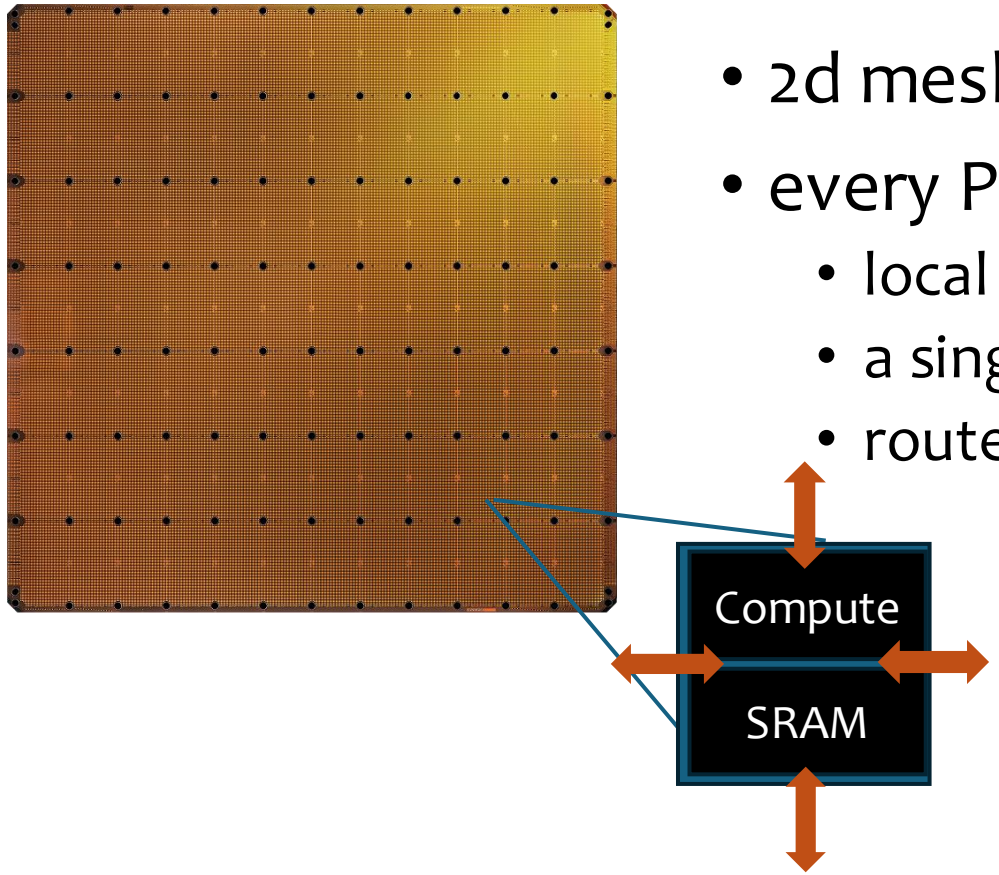
The Wafer-Scale Engine



- contains over 4 trillion transistors (Nvidia's B200 has 208 million)
- draws roughly twice the power of an 8 GPU node

- 900,000 cores w/ on-chip fabric interconnect
- "designed for the fine-grained dynamic sparsity in neural networks" [WSE3 whitepaper]

Hardware Model



- 2d mesh of processing elements (PEs)
- every PE features:
  - local SRAM
  - a single-threaded general-purpose core
  - routers to each of four neighbors

Programming Model

- We use a new low-level programming model (language + compiler) in development by Cerebras
- Communication and computation are separate…

Computation
Tungsten: a C-like language for describing compute within a PE

Communication
Paint: an orchestration language for describing router states, communication patterns, and PE layouts

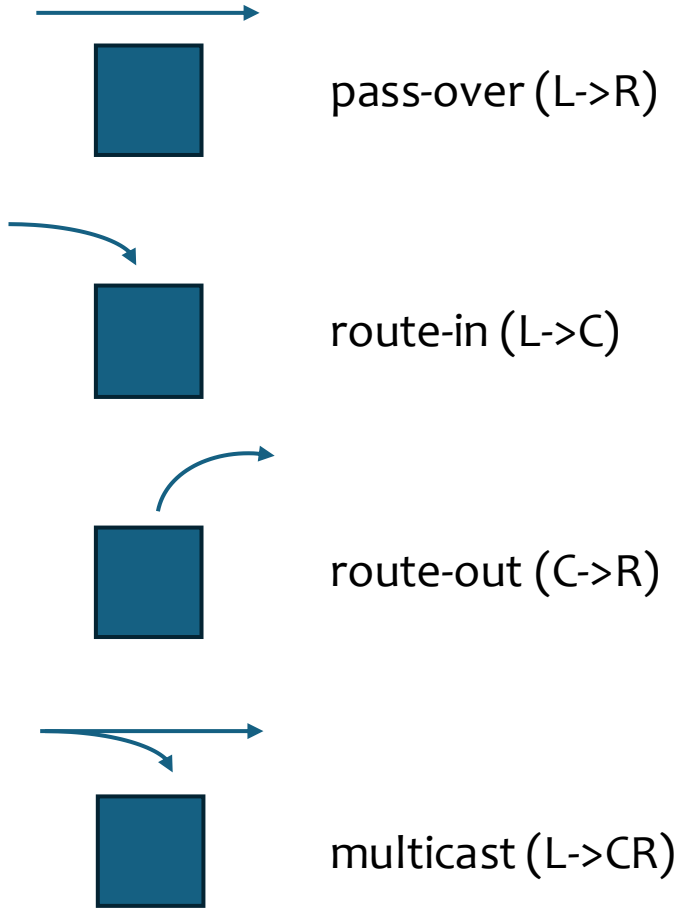
Computation on the WSE

- No caches, or related overheads
- Every PE has…
  - 48KB local SRAM (wafer supports peak of 21PB/sec bidirectional BW)
  - general purpose compute and SIMD support for tensor operations

	single PE achieved Flops	single PE achieved BW	wafer achieved BW
copy	-	7.29 GB/sec	6.56 PB/sec
scale	.658 Gflops/sec	5.26 GB/sec	4.74 PB/sec
sum	.632 Gflops/sec	5.05 GB/sec	4.55 PB/sec
triad	1.09 Gflops/sec	6.56 GB/sec	5.91 PB/sec

Communication on the WSE

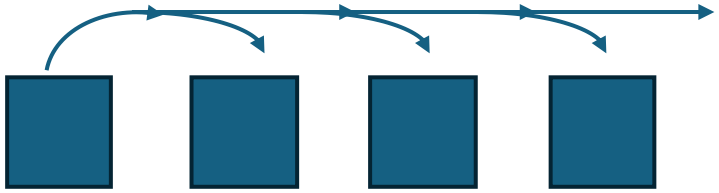
- Communication between PEs is organized in virtual channels, called “colors”
- At every PE, the routers are configured separately for every color
- The codes L,R,U, and D refer to each of a PE’s four neighbors.
- The code C routes data into the compute engine of the PE.



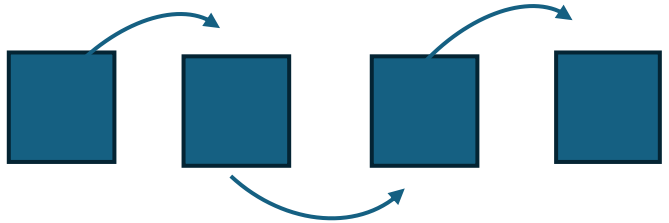
1D Collectives

Luczynski et al. explore 1D and 2D collectives on the WSE

- 1D broadcast:
  chain of multicasts



- 1D reduce:
  chain of partial reductions

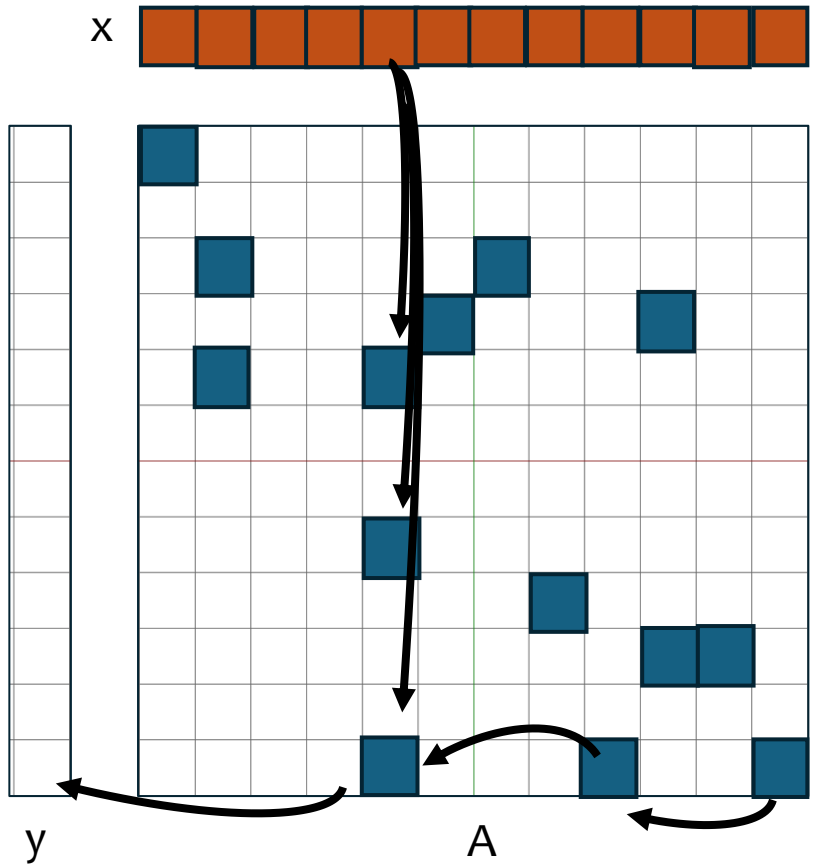


P. Luczynski, L. Gianinazzi, P. Iff, L. Wilson, D. De Sensi, and T. Hoefler, "Near-optimal wafer-scale reduce," HPDC '24.

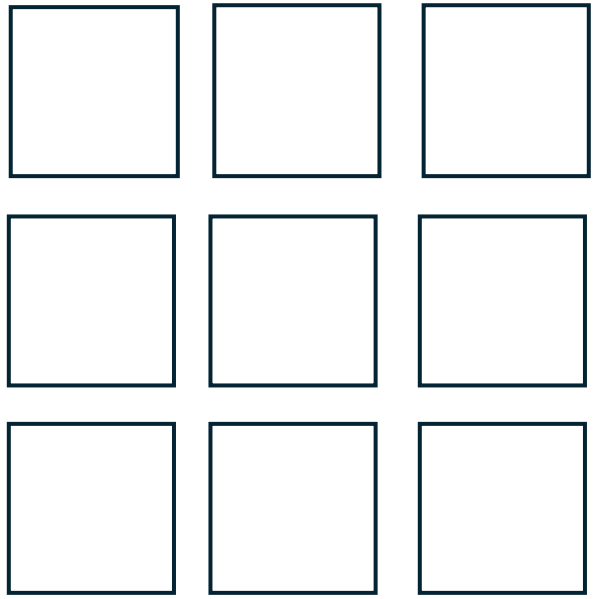
The intuition for SpMV

Given a dense input vector x and sparse A…

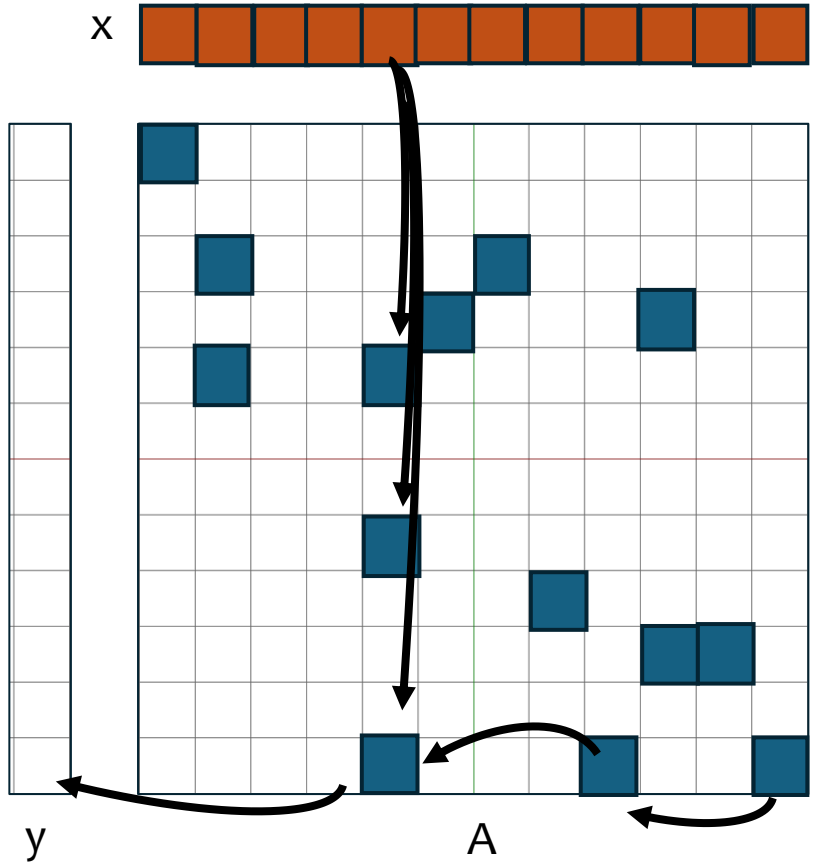
1. Broadcast x values down columns to non-zeros of A
2. Compute local partial products
3. Reduce across rows to accumulate results to the left



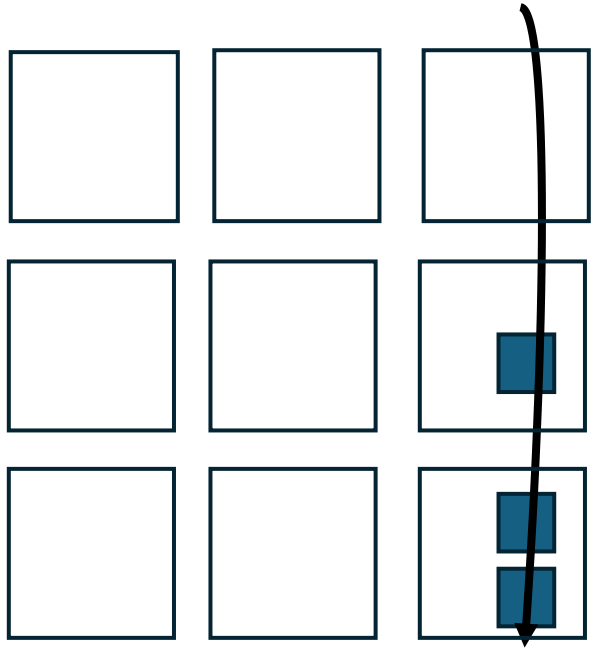
Sparse Layout of A



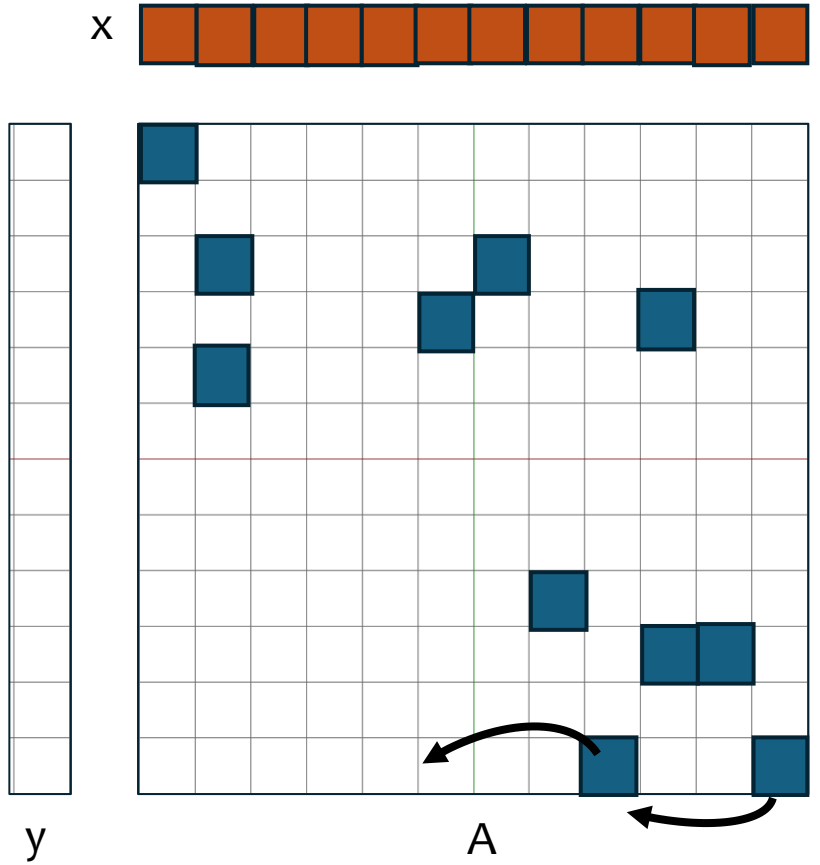
3x3 grid of PEs



Sparse Layout of A

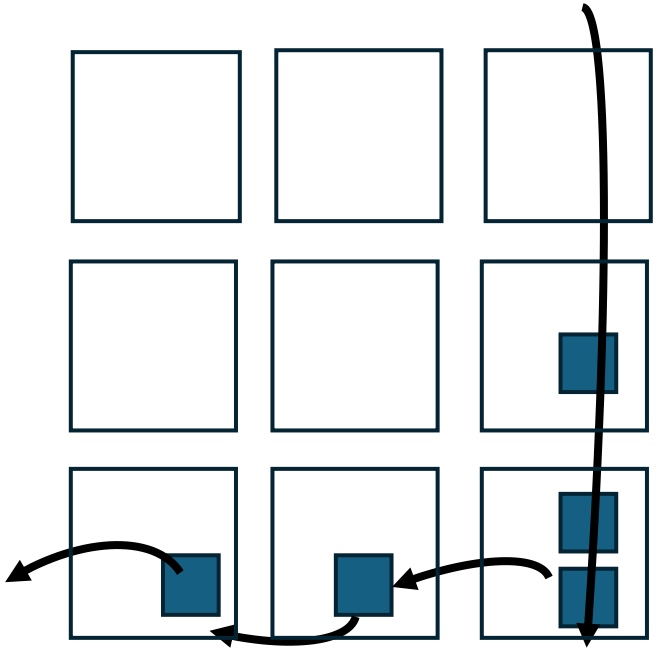


3x3 grid of PEs

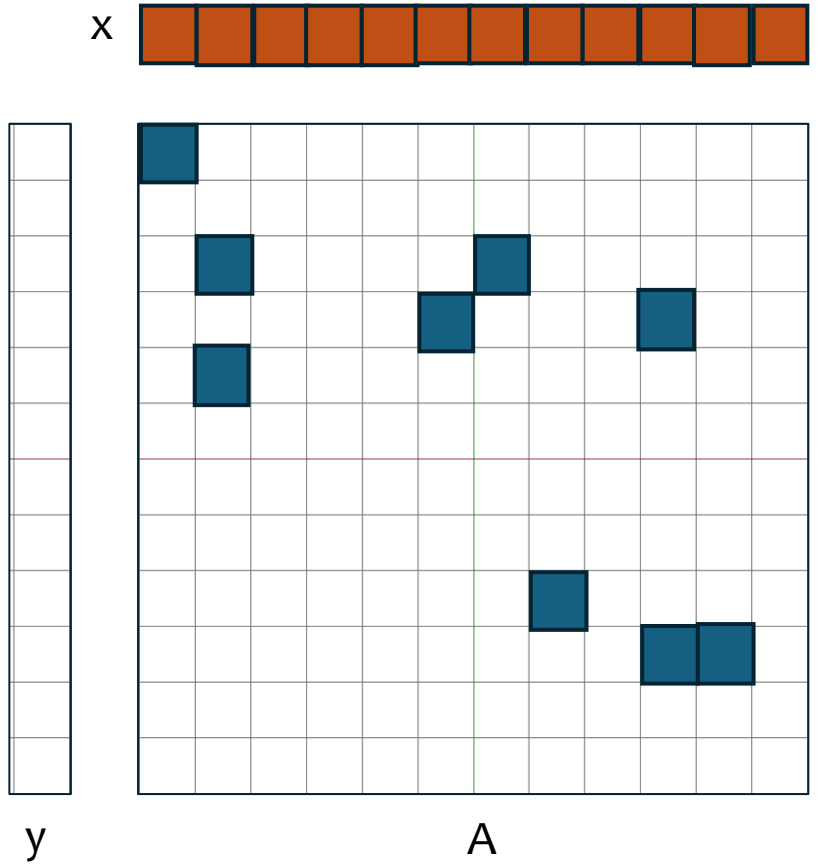


Sparse Layout of A

"Row- and Column-Locality Preserving"

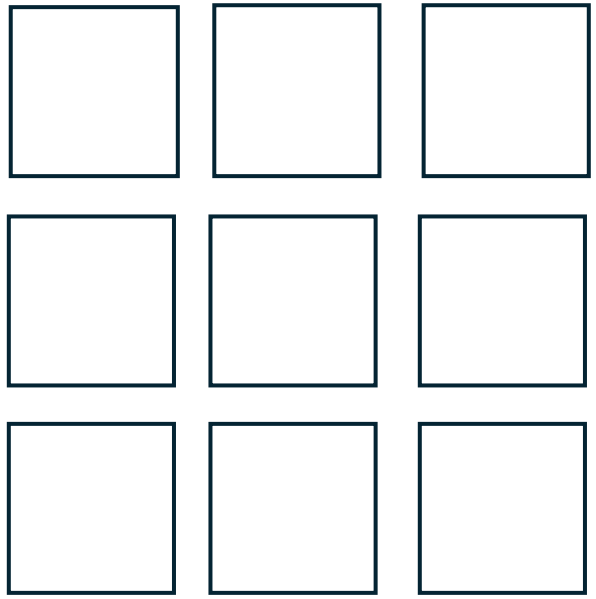


3x3 grid of PEs

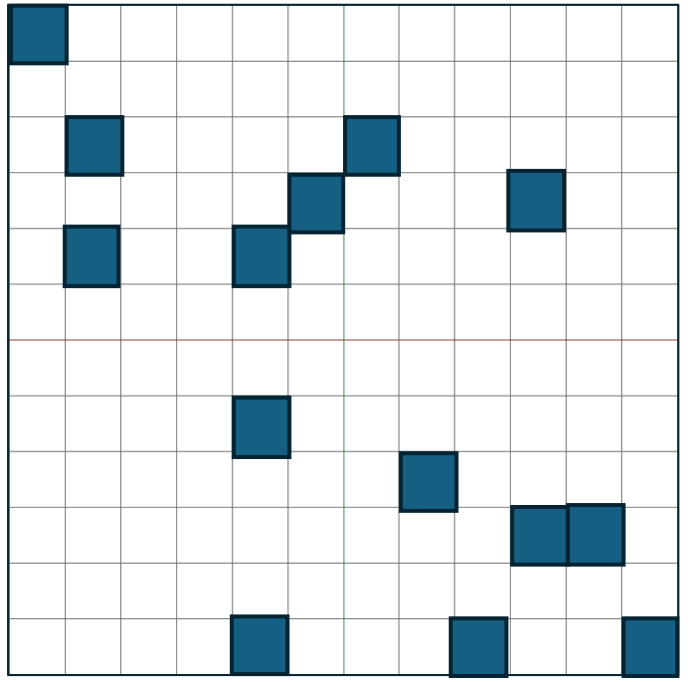


Sparse Layout of A

"Row- and Column-Locality Preserving"

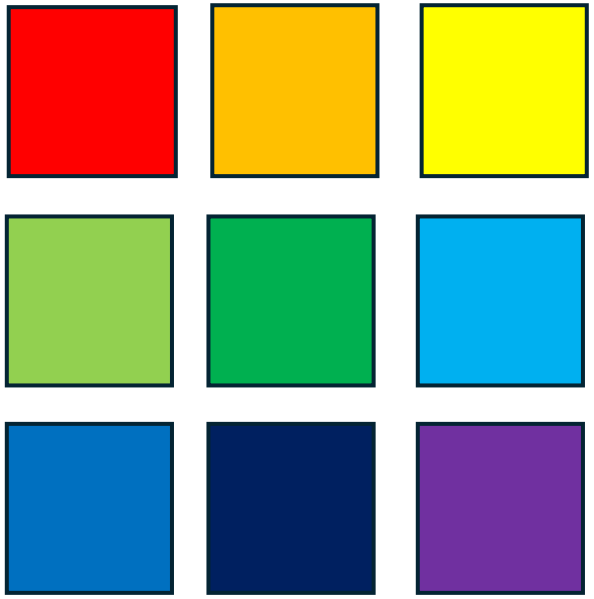


3x3 grid of PEs



Sparse Layout of A

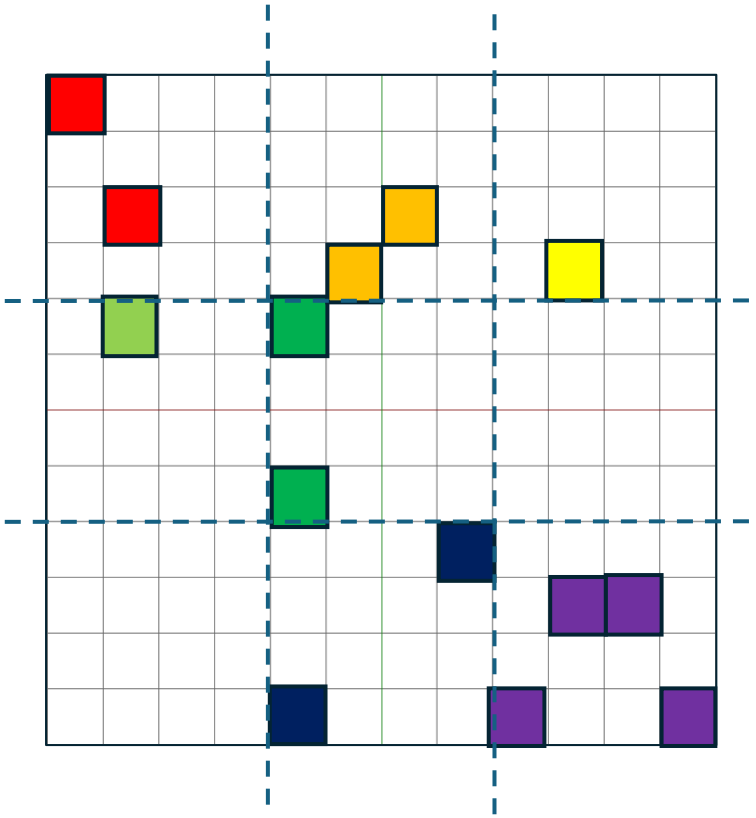
"Row- and Column-Locality Preserving"



3x3 grid of PEs

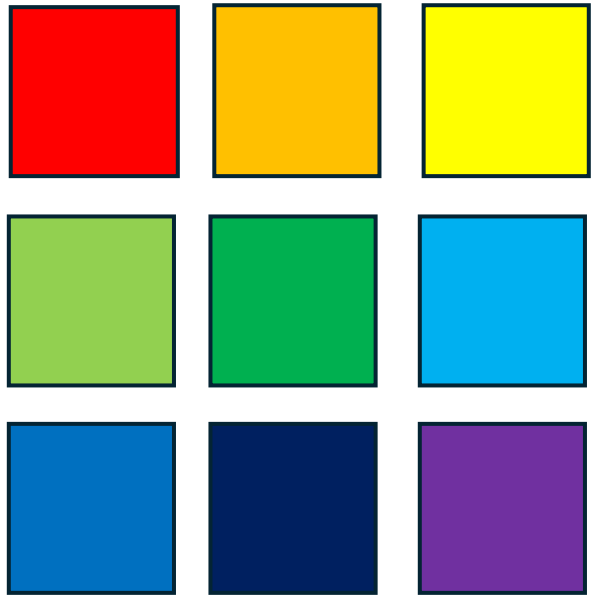
2D-block mapping

$$A_{ij} \rightarrow PE_{\left\{\frac{i}{4'}\frac{j}{4}\right\}}$$



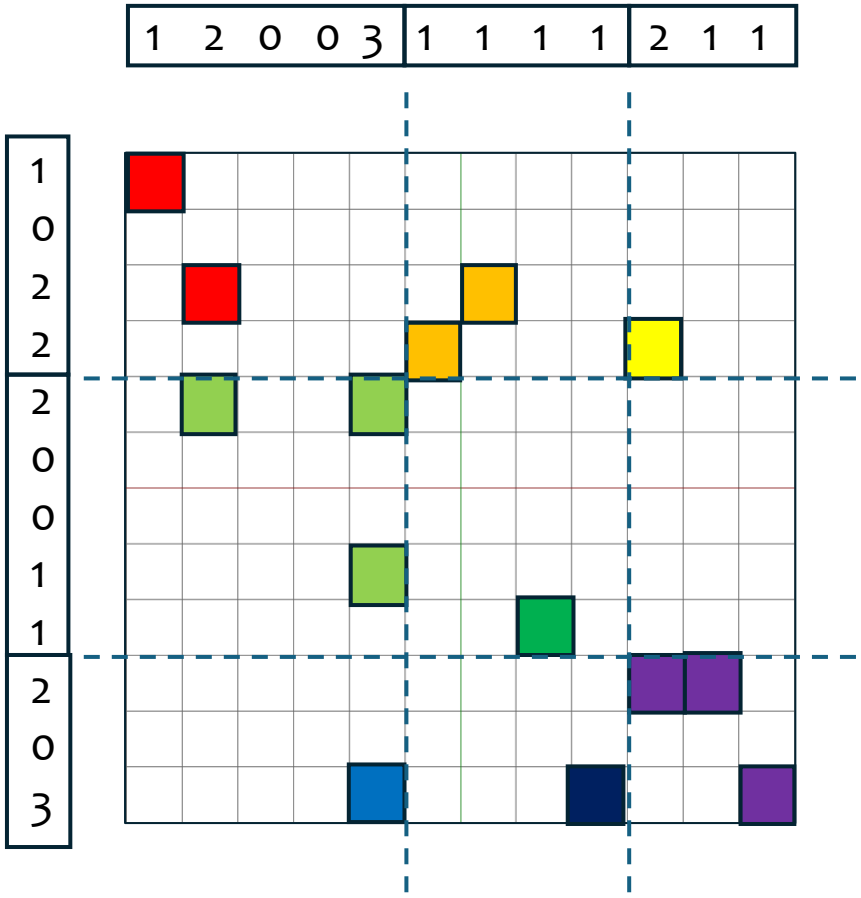
Sparse Layout of A

"Row- and Column-Locality Preserving"



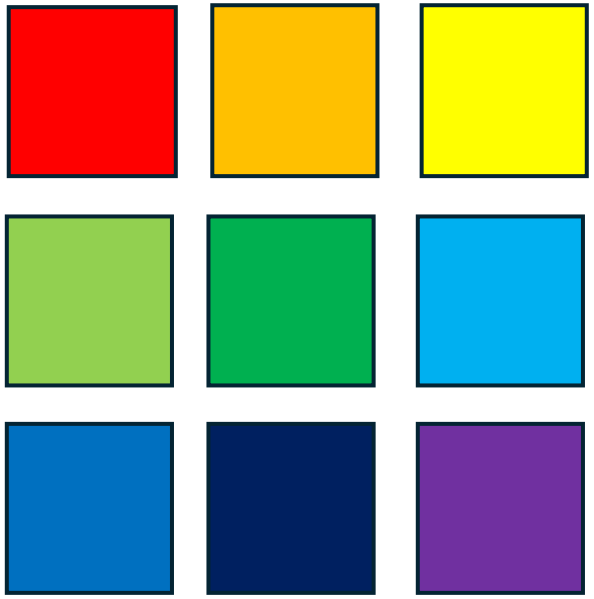
3x3 grid of PEs

2D-block mapping
with sparsity-awareness



Sparse Layout of A

"Row- and Column-Locality Preserving"

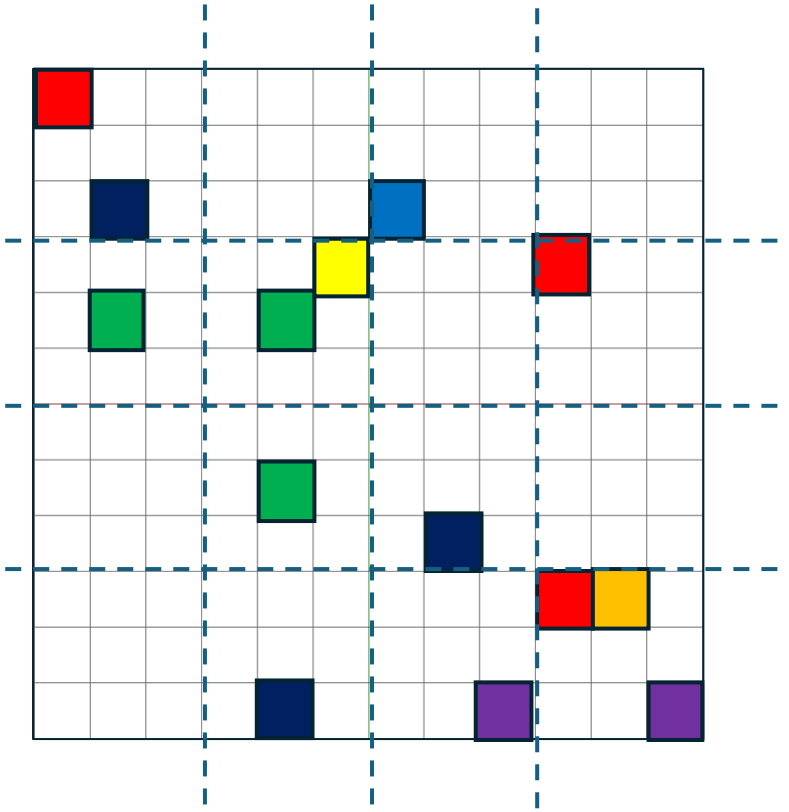


3x3 grid of PEs

2D-cyclic mapping

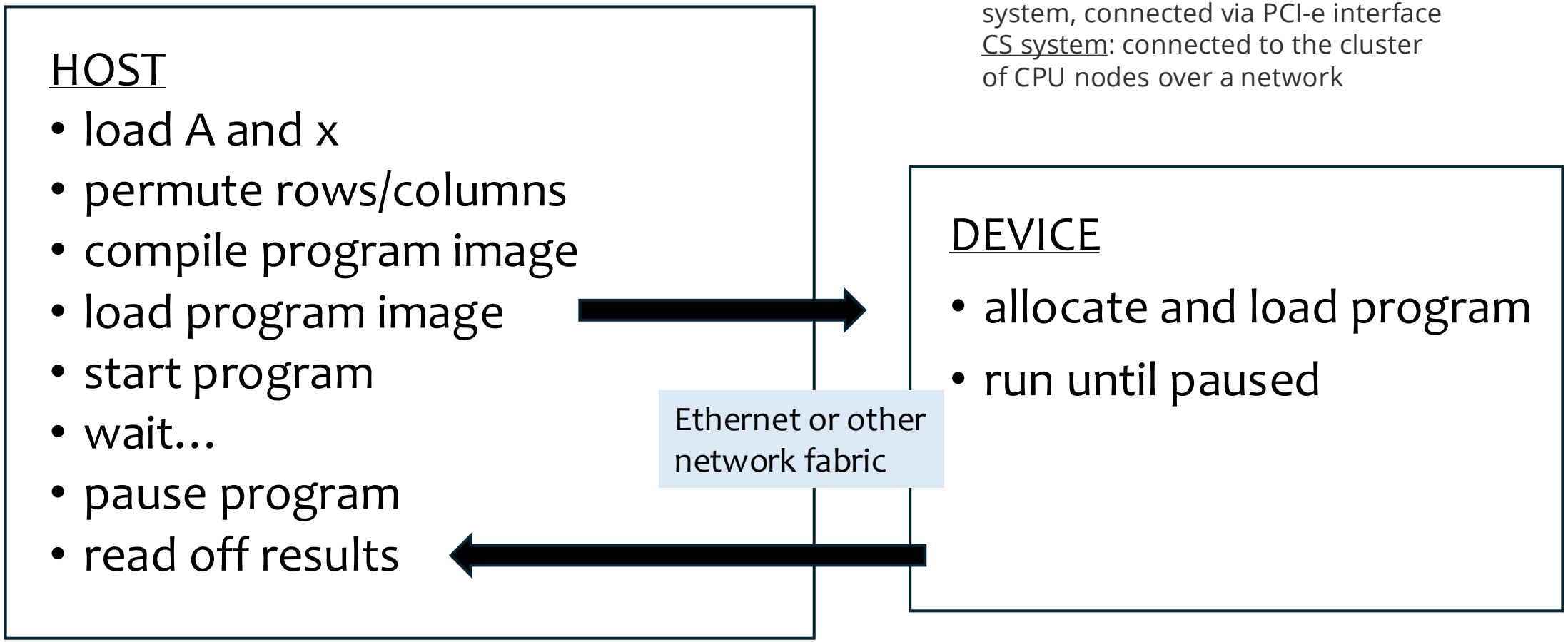
$A_{ij} \rightarrow PE_{\{i \bmod 3, j \bmod 3\}}$

within a PE, non-zeros are stored in CSR



Running SpMV on the WSE

conventional GPU: co-located with a host CPU in the same physical system, connected via PCI-e interface
CS system: connected to the cluster of CPU nodes over a network



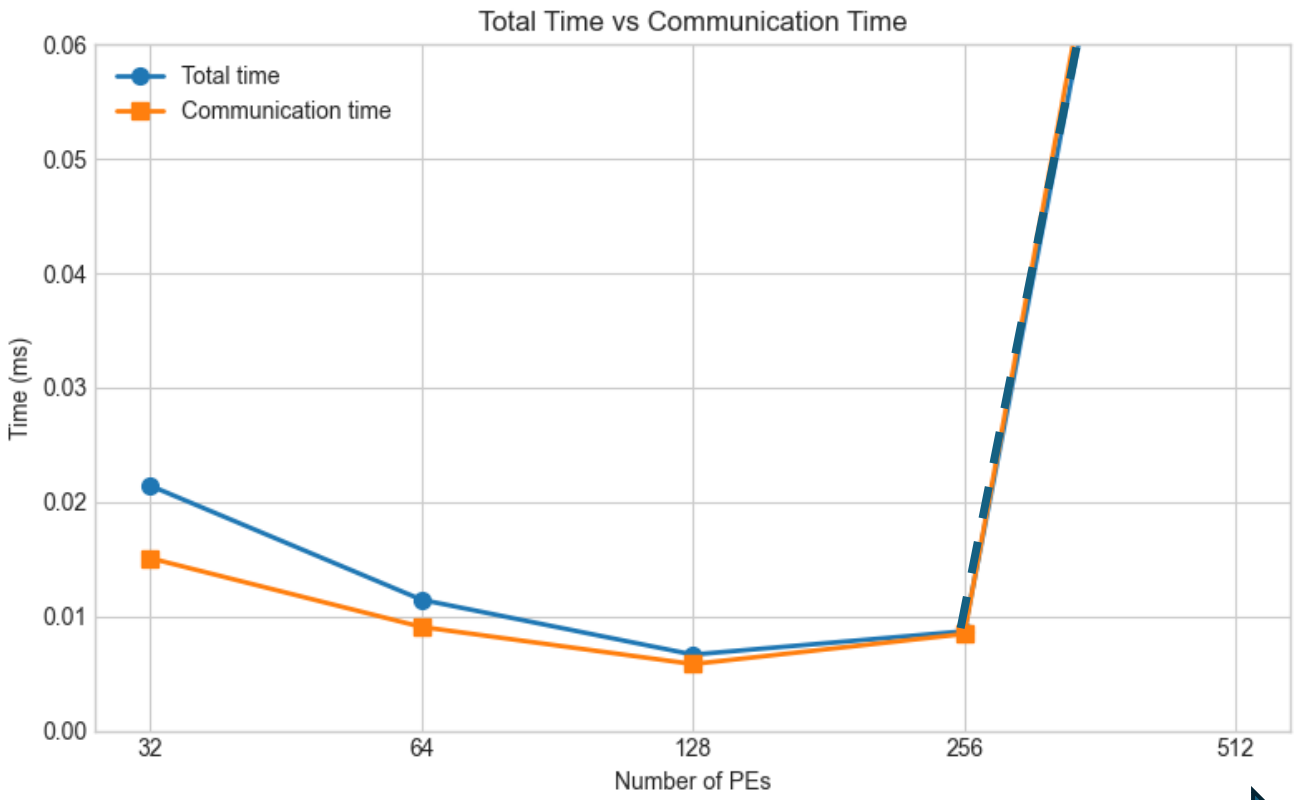
Evaluation – Random NNZ Distribution

A: 2048x2048

- with 6.25% sparsity

Strong Scaling:

- on square PE grids



64 matrix rows per PE row 4

256 nonzeros per PE 1

Evaluation – Random NNZ Distribution

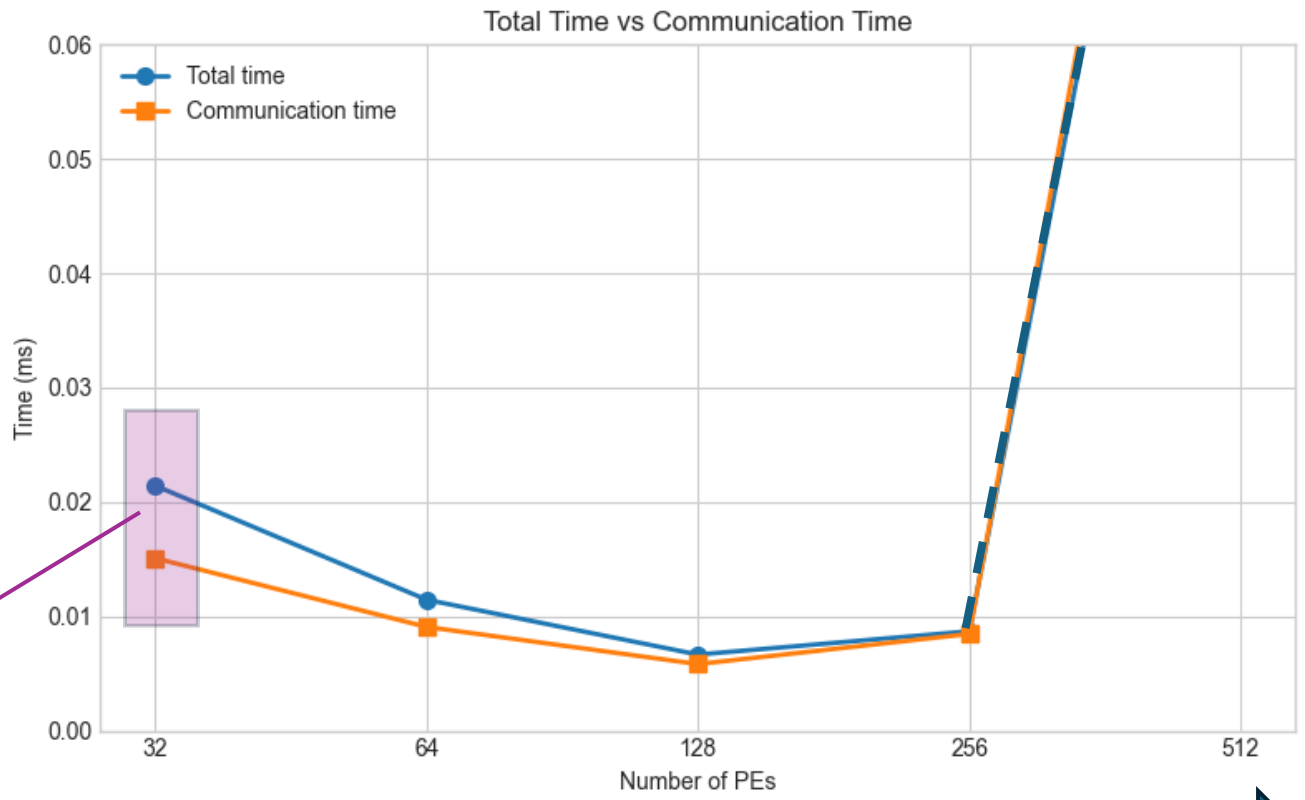
A: 2048x2048

- with 6.25% sparsity

Strong Scaling:

- on square PE grids

component	cycle count
total	23579
comm only	16596
comm components	5302



64 matrix rows per PE row 4

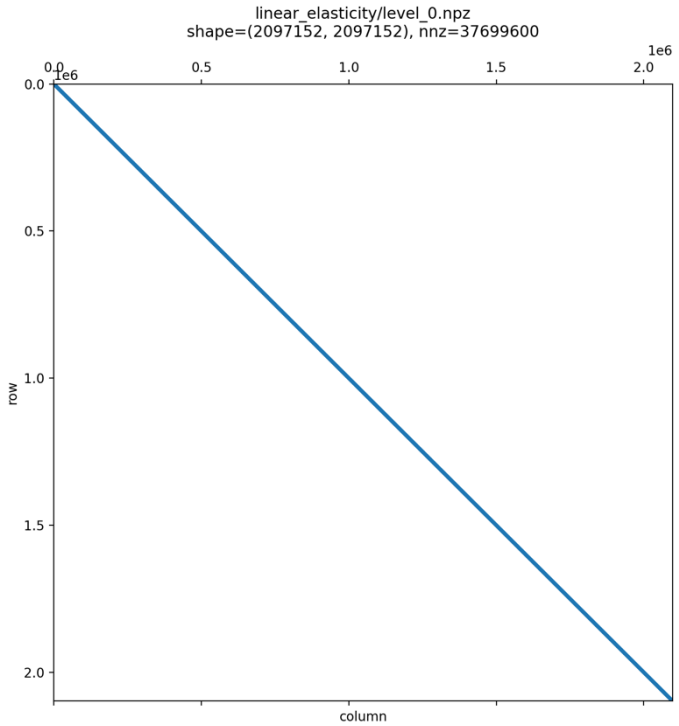
256 nonzeros per PE 1

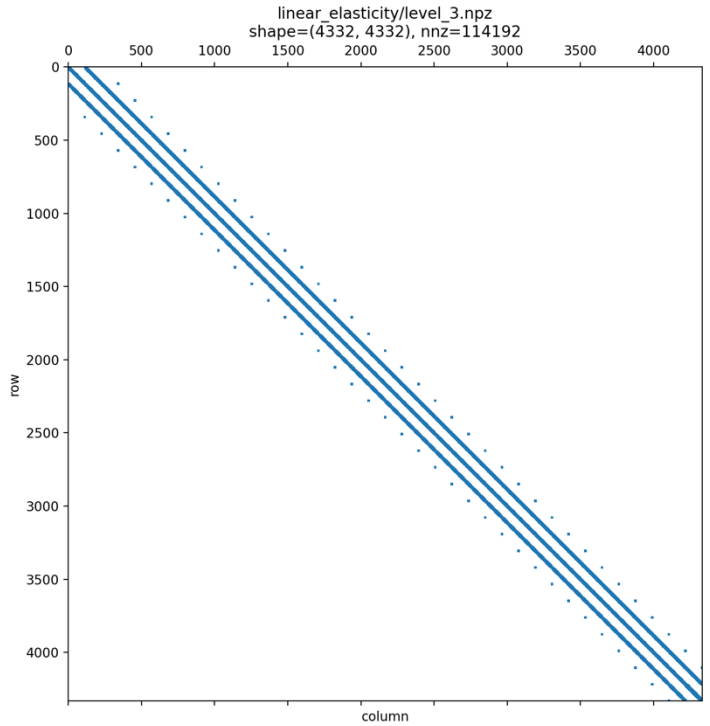
Evaluation – Linear Elasticity

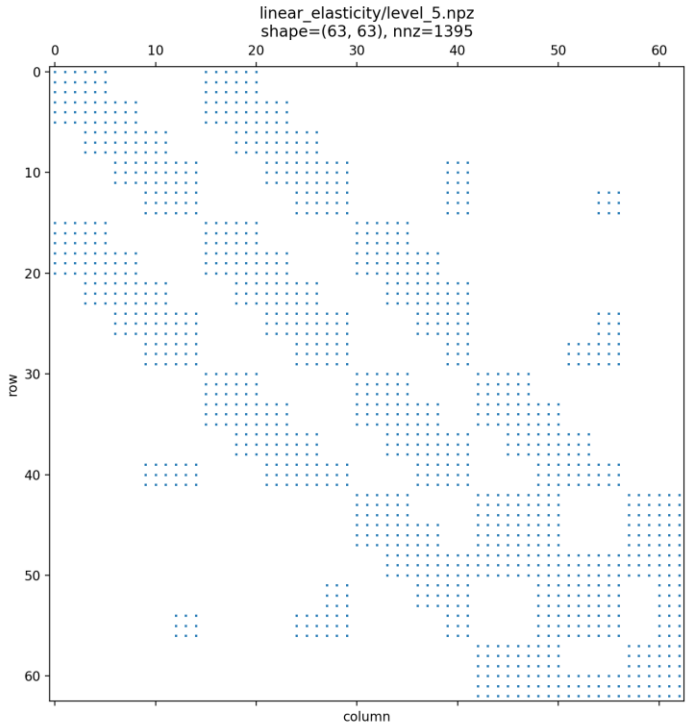
- Generated a 6-level AMG hierarchy
  - from pyAMG gallery.linear_elasticity
  - using smoothed aggregation
- Evaluating just SpMV (random RHS)
- How do various architectures scale?

Lvl	Rows	NNZ
0	2,097,152	37,699,600
1	350,892	9,437,184
2	38,988	1,044,432
3	4,332	114,192
4	507	12,717
5	63	1,395

Evaluation – Linear Elasticity

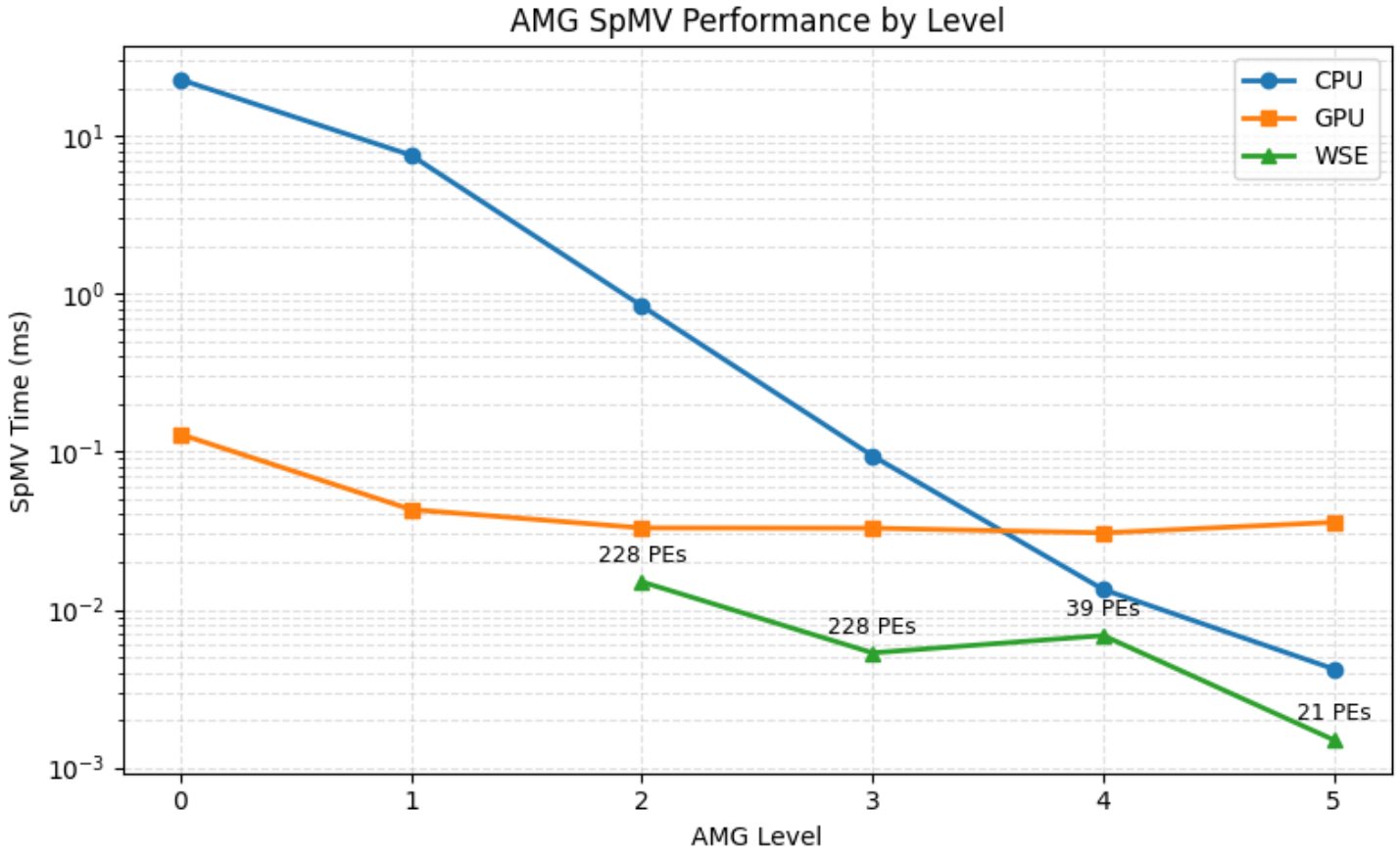






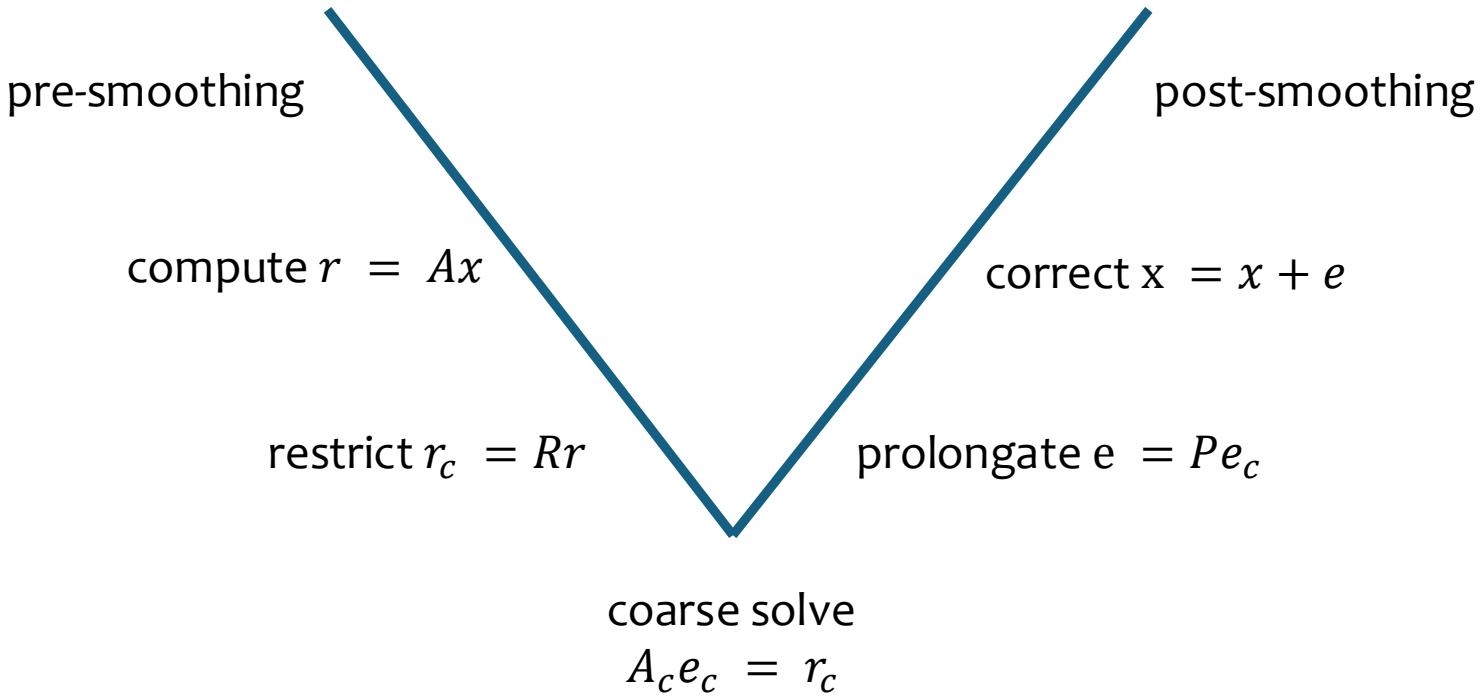
Evaluation – Linear Elasticity

- GPU: Nvidia H200
  - cuSPARSE impl
- WSE difficulties:
  - memory capacity
  - naïve partitioning forces even division
- But! Potential for good scaling even at coarsest level.



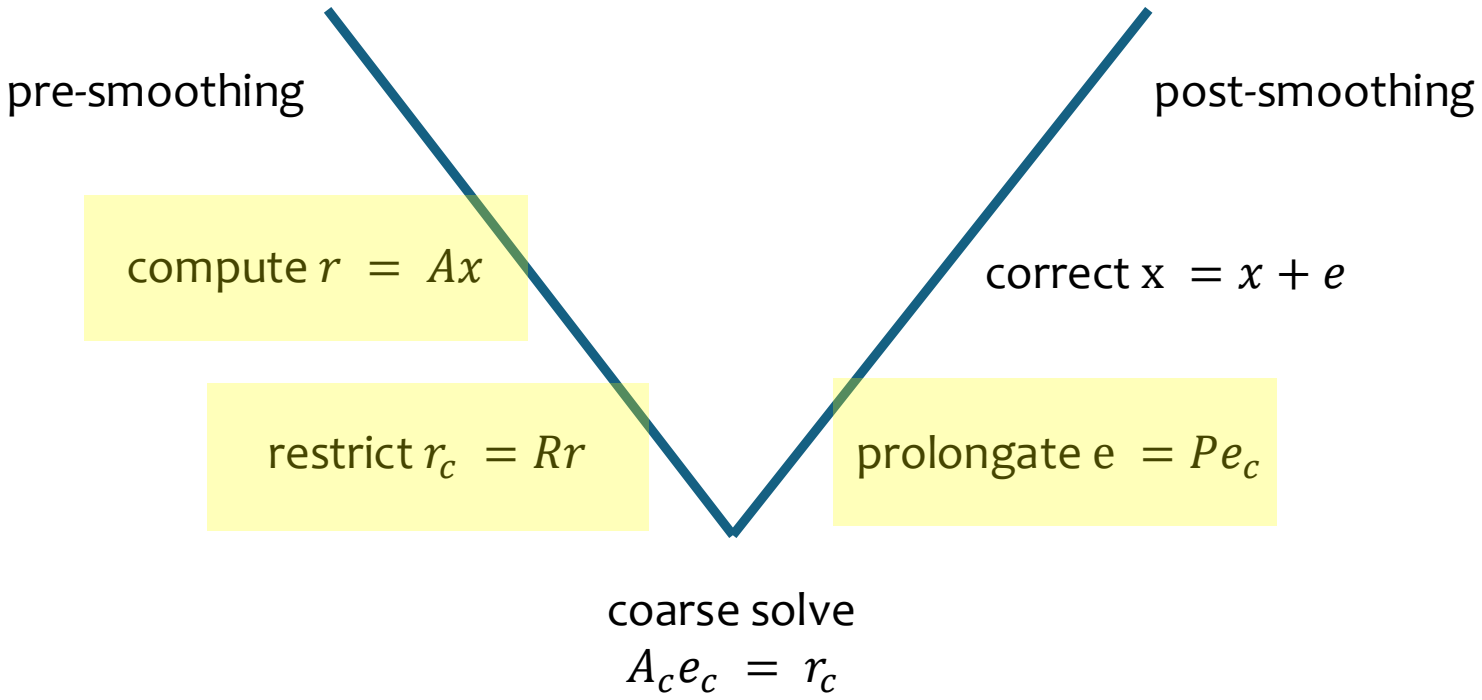
SpMV in the AMG Cycle

2-level AMG V-cycle



SpMV in the AMG Cycle

2-level AMG V-cycle



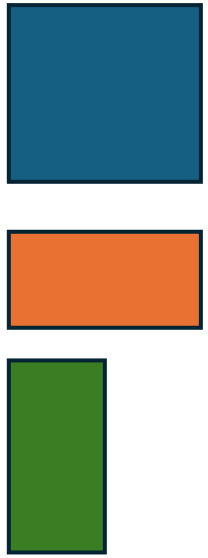
SpMV in the AMG Cycle

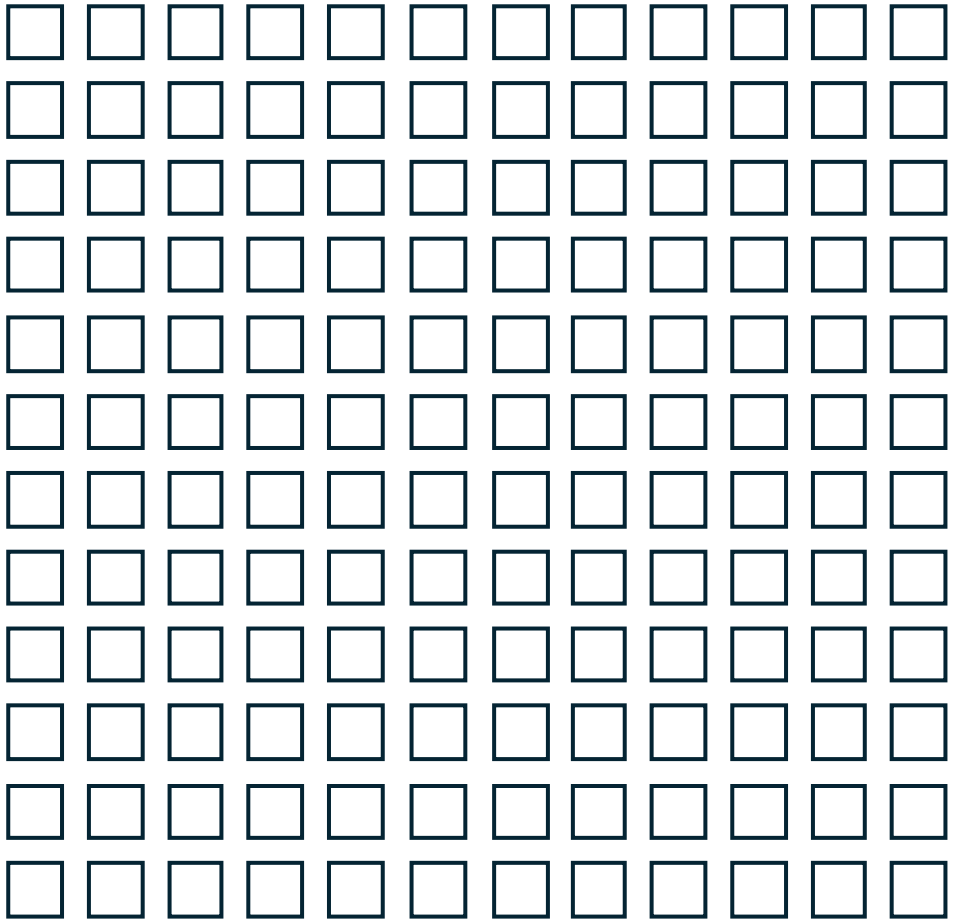
2-level AMG V-cycle

compute $r = Ax$

restrict $r_c = Rr$

prolongate e $= Pe_c$





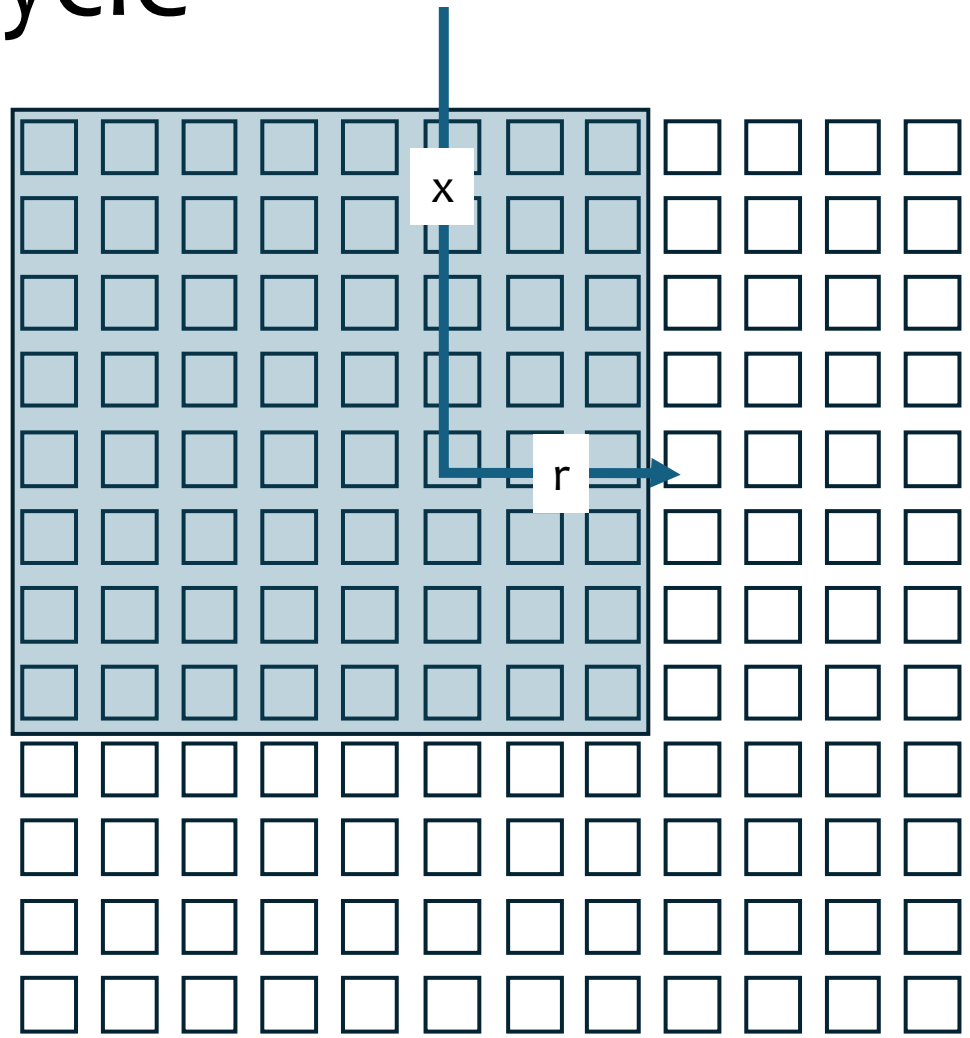
SpMV in the AMG Cycle

2-level AMG V-cycle

compute $r = Ax$

restrict $r_c = Rr$

prolongate e $= Pe_c$



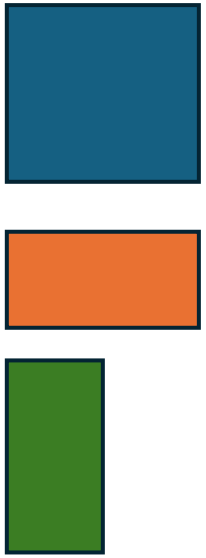
SpMV in the AMG Cycle

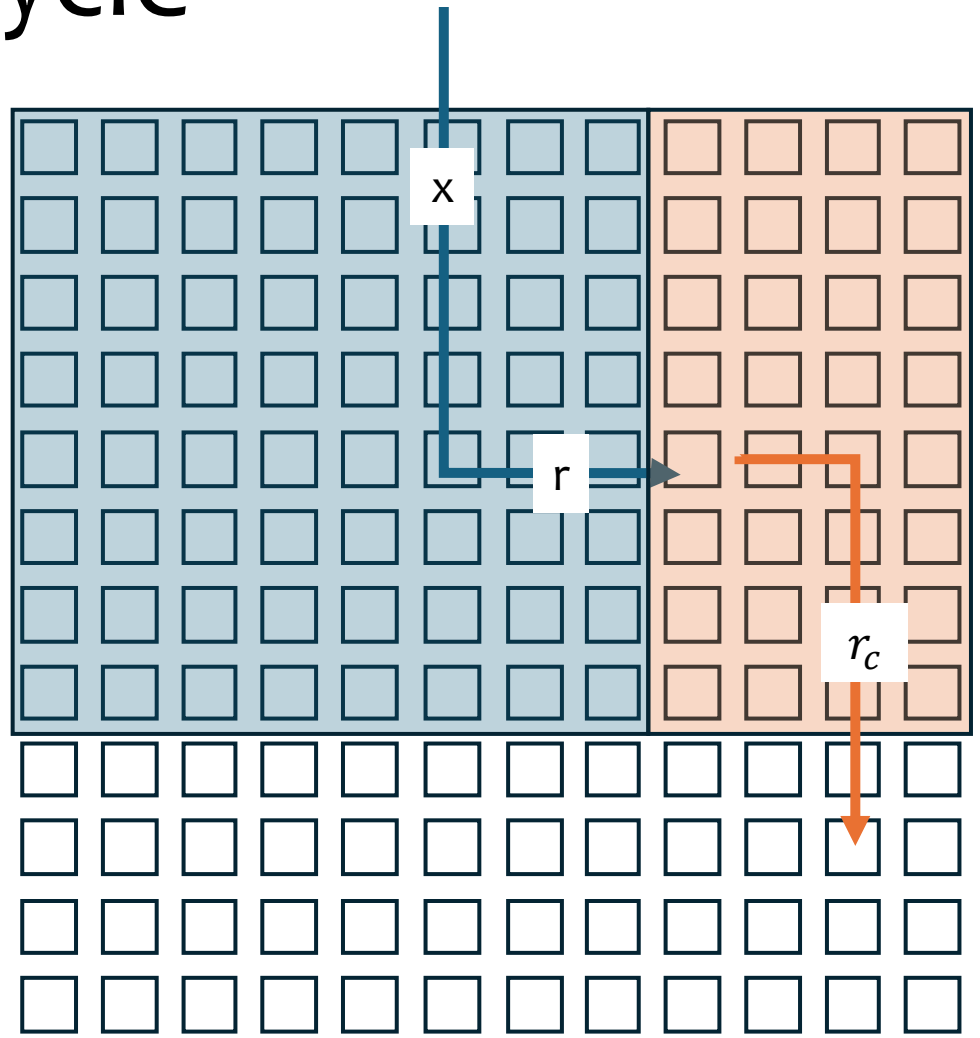
2-level AMG V-cycle

compute $r = Ax$

restrict $r_c = Rr$

prolongate e $= Pe_c$





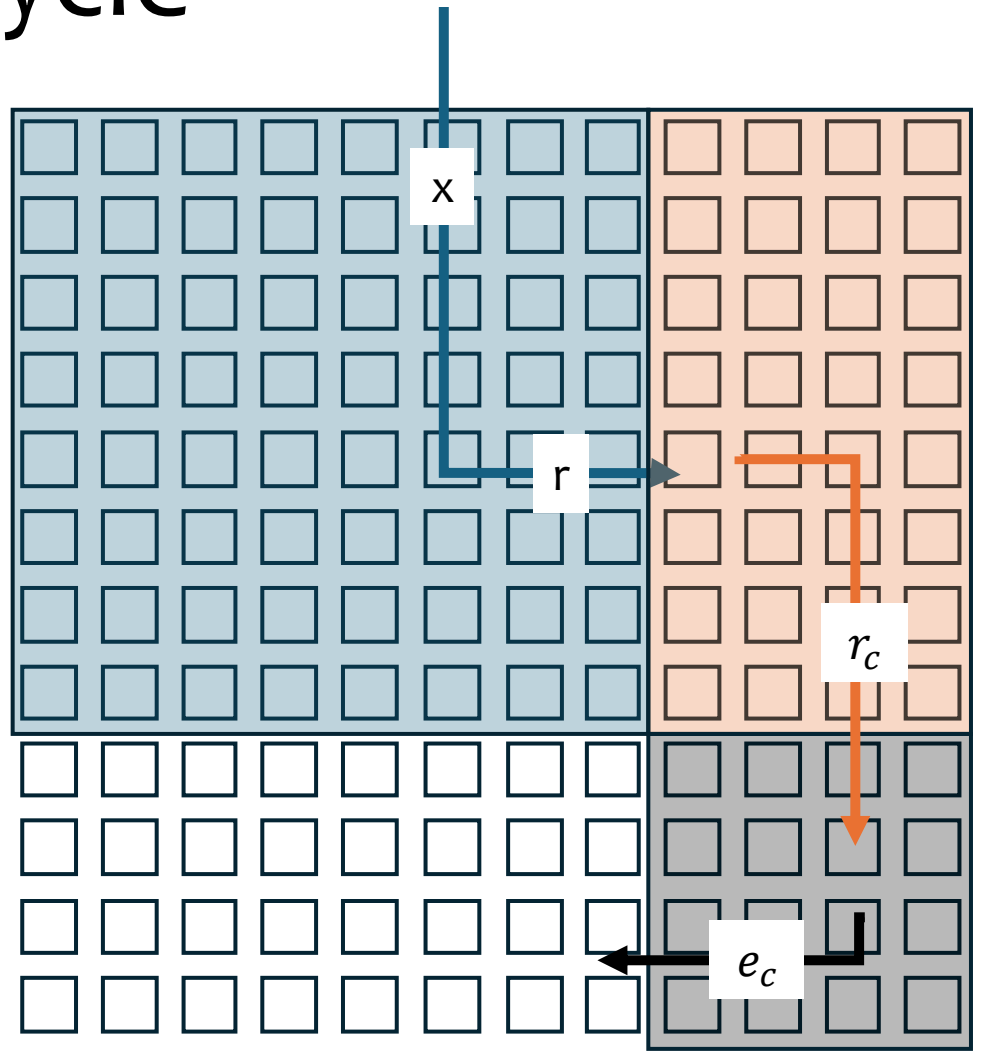
SpMV in the AMG Cycle

2-level AMG V-cycle

compute $r = Ax$

restrict $r_c = Rr$

prolongate e $= Pe_c$



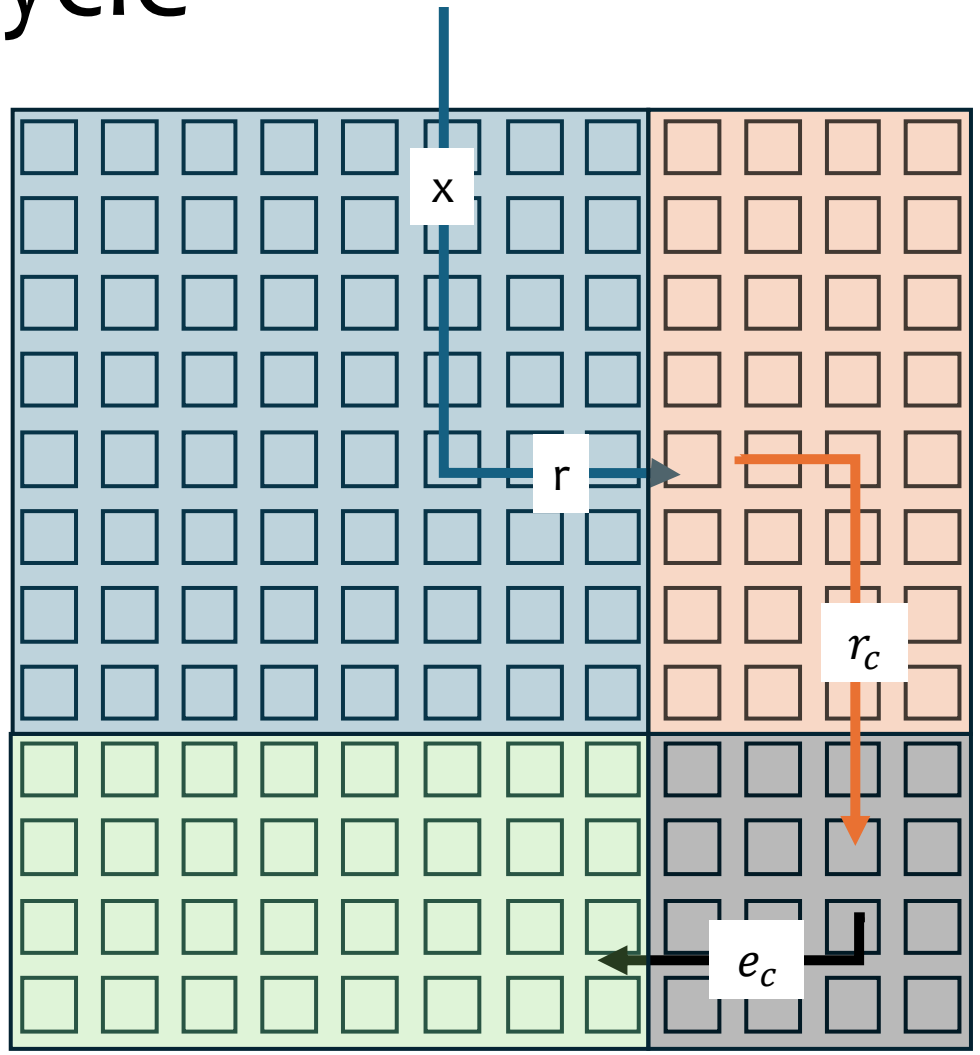
SpMV in the AMG Cycle

2-level AMG V-cycle

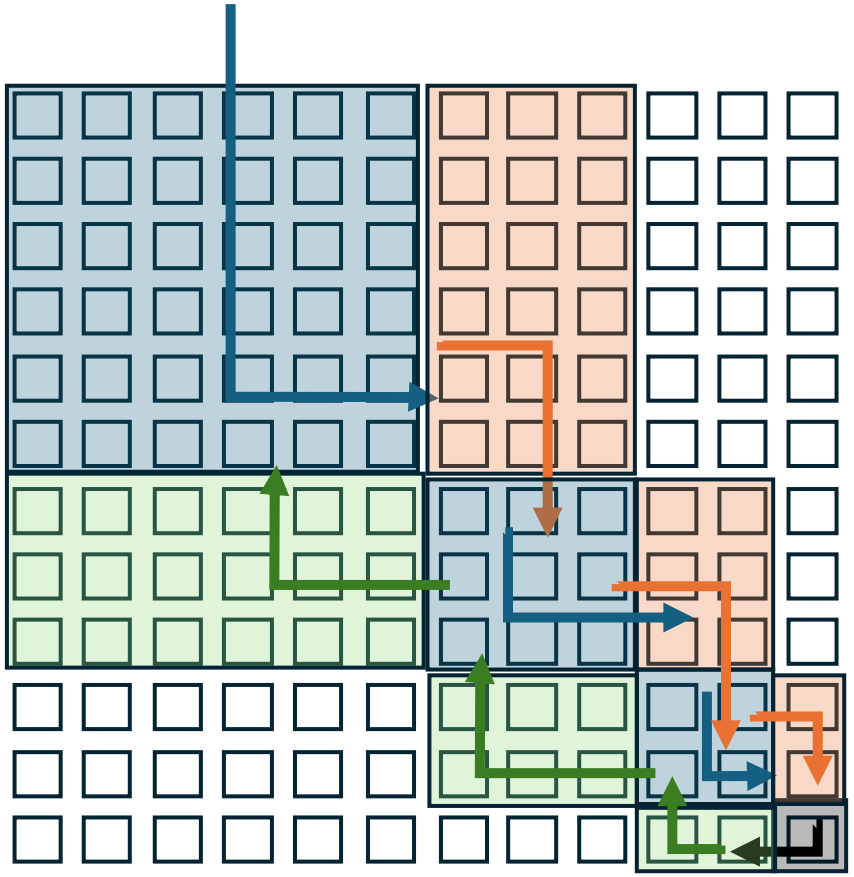
compute $r = Ax$

restrict $r_c = Rr$

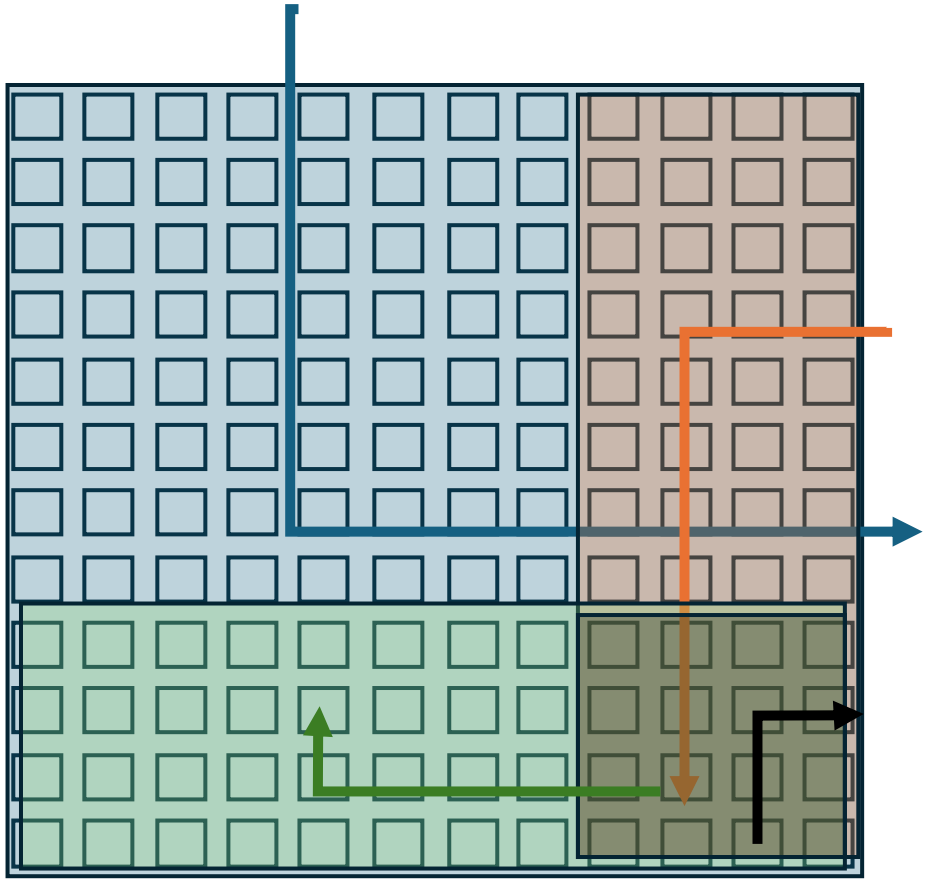
prolongate e $= Pe_c$



AMG Cycle Layouts



Recursive layout



Nested layout

Next steps

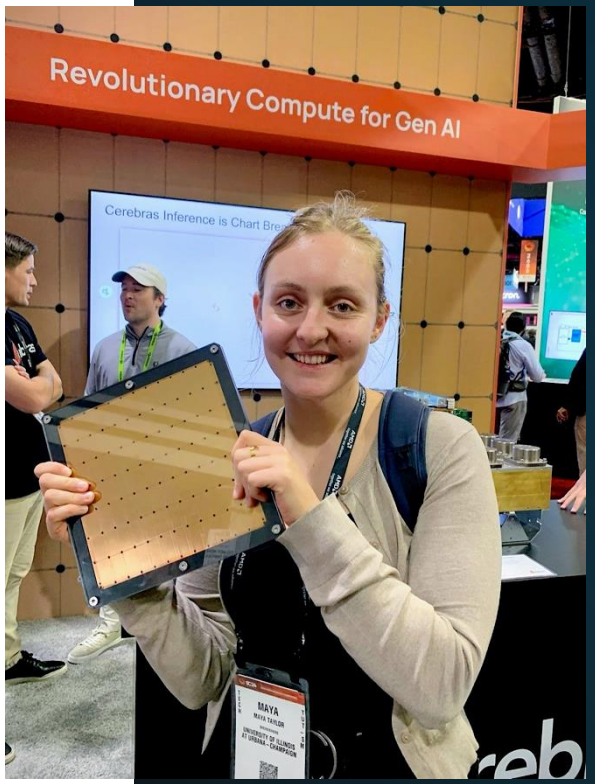
- SpMV implementation:
  - Performance analysis
  - Understanding of runtime breakdown
  - Address issues around:
    - blowup
    - communication decomposition
    - load balance of non-zeros
- AMG:
  - Compose multiple SpMVs
  - Analysis of various cycle layouts
  - Smoother and solve implementations

Thank you!

contact: mayat4@illinois.edu

This material is based in part upon work supported by the Department of Energy, National Nuclear Security Administration under Award Number DE-NA0003963

and 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.



Me at SC'24!