

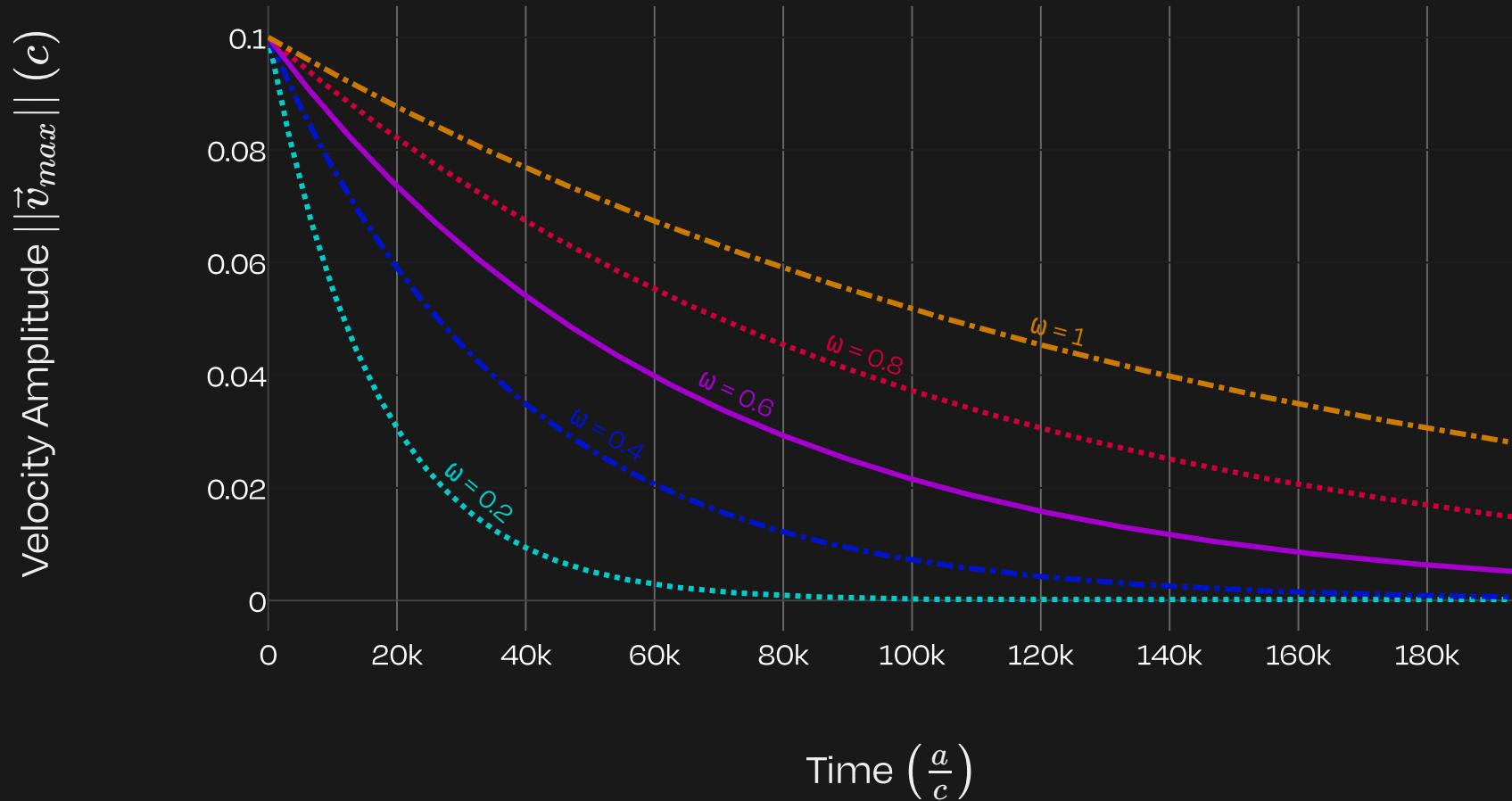
The Lattice Boltzmann Method in Kokkos

Performance Lessons Learned

Julian Karrer

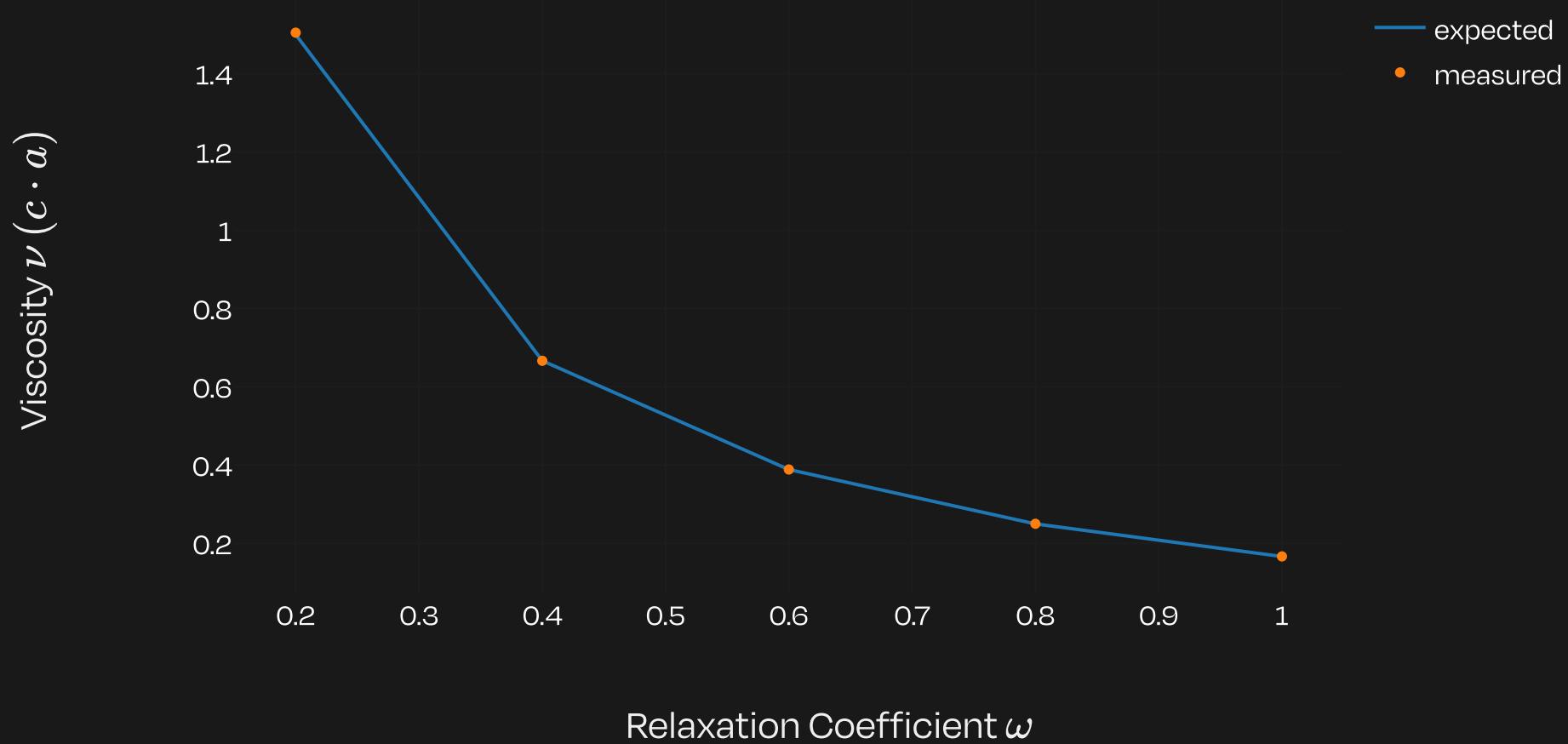


Shearwave decay



200k time steps of D2Q9 shearwave decay with $u_x(t = 0) = 0.1, \omega \in \{k \in \mathbb{N}_1^5 \mid \frac{1}{k}\}$ on 1000×1000 nodes

Shearwave decay



200k time steps of D2Q9 shearwave decay with $u_x(t = 0) = 0.1, \omega \in \{k \in \mathbb{N}_1^5 \mid \frac{1}{k}\}$ on 1000×1000 nodes

1. Hand-Tuning and Precision

Micro-optimizations

1. Manual Loop-Unrolling
2. Common Subexpression Elimination
3. Arithmetic Simplification
4. Read/Write reordering

NO significant difference ($\pm\sigma$) for any tested GPU

Precision

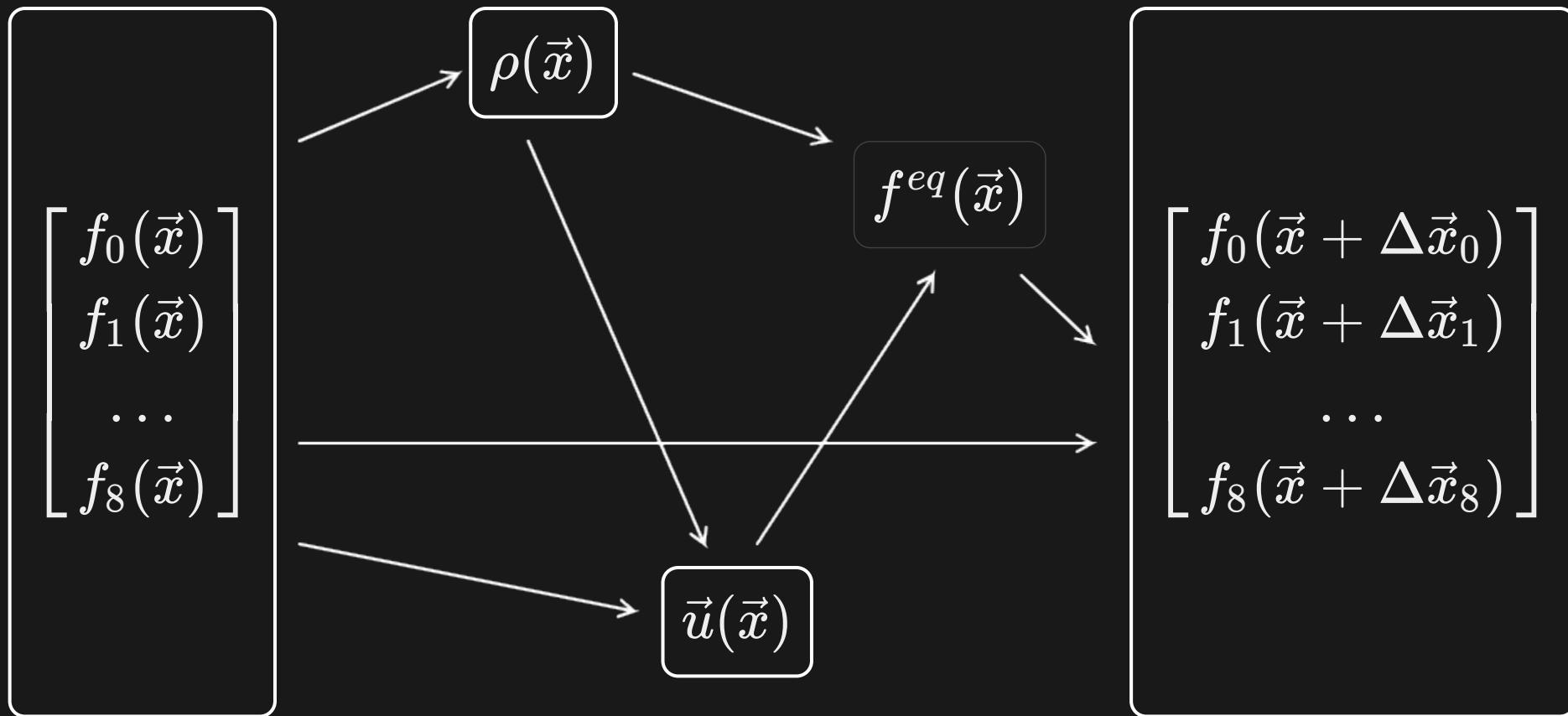
- Use **float** instead of **double** if possible!

GPU	relative Speedup for float
3060Ti	+102%
MI300A	+100%
H100	+101%

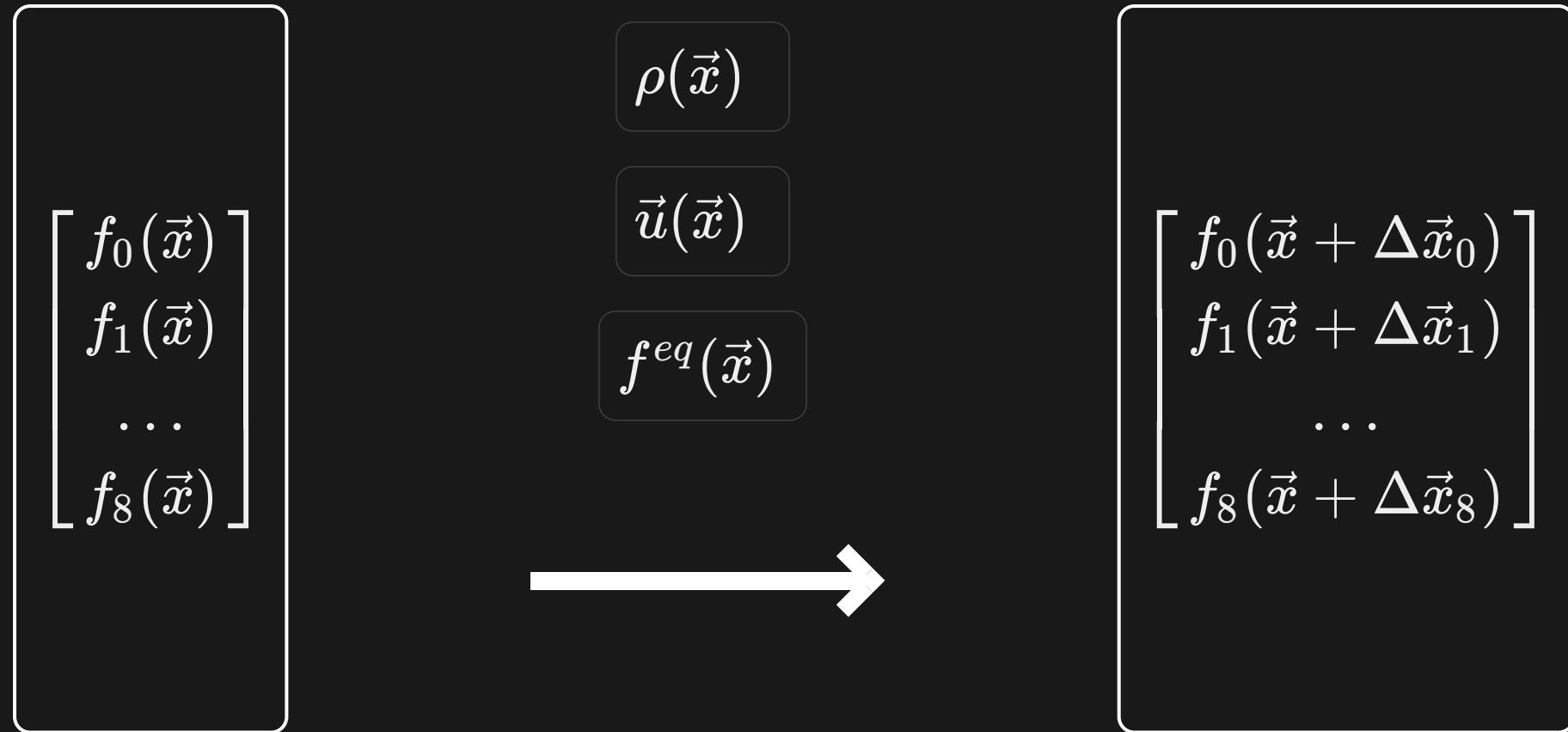
- Factor of two for variety of hardware

2. Efficiently use Memory Bandwidth

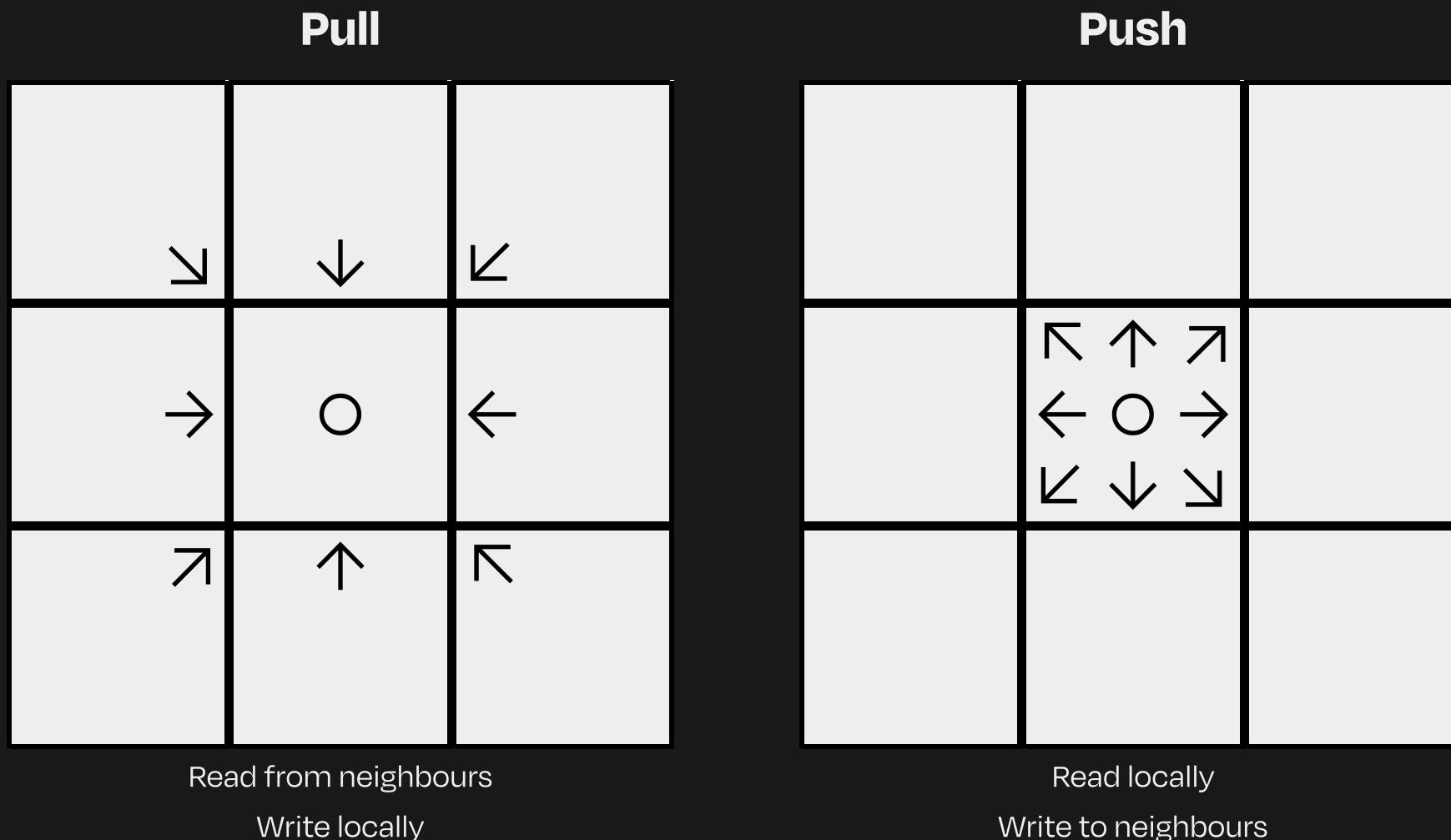
Data Dependencies



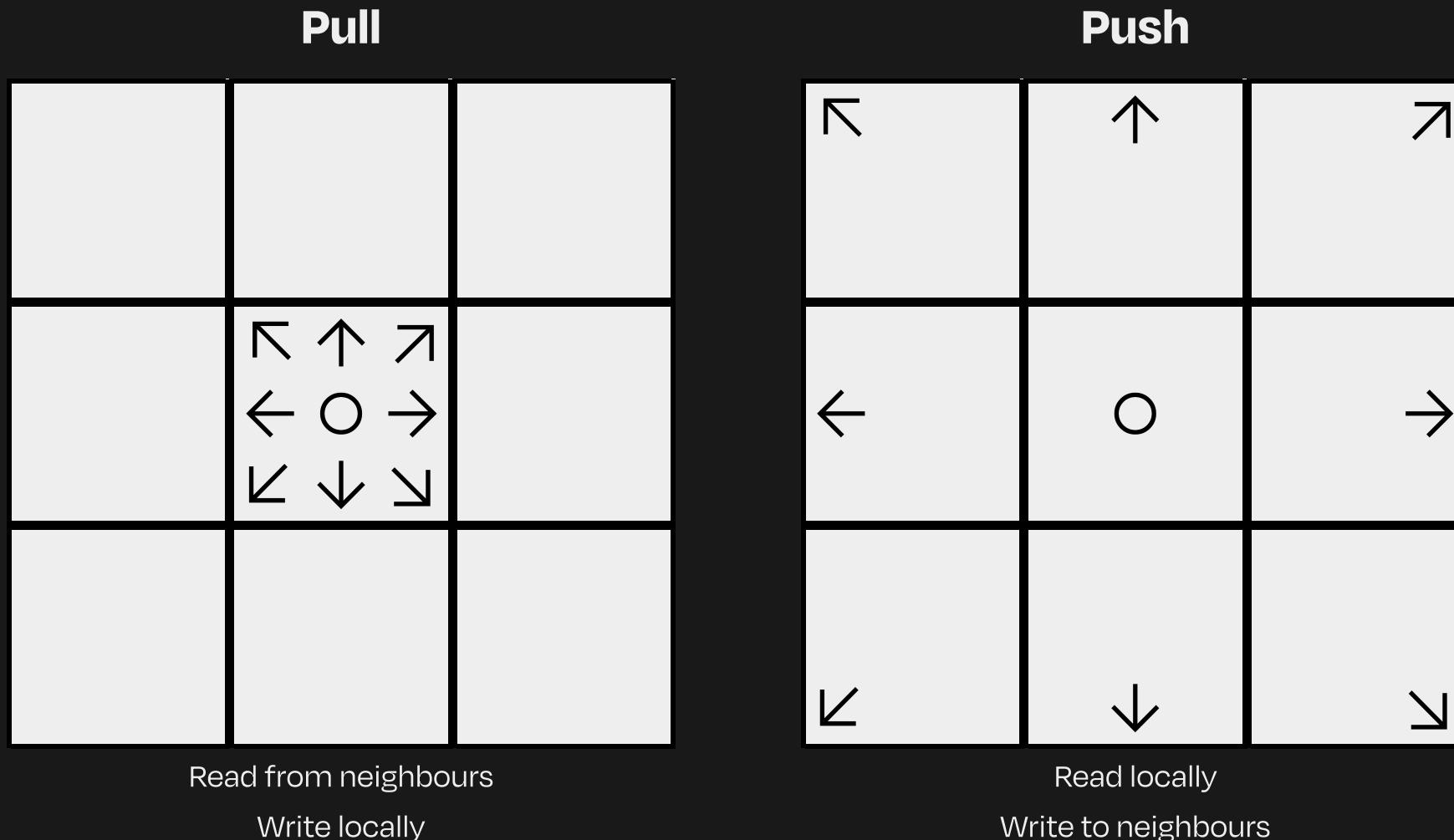
Data Dependencies



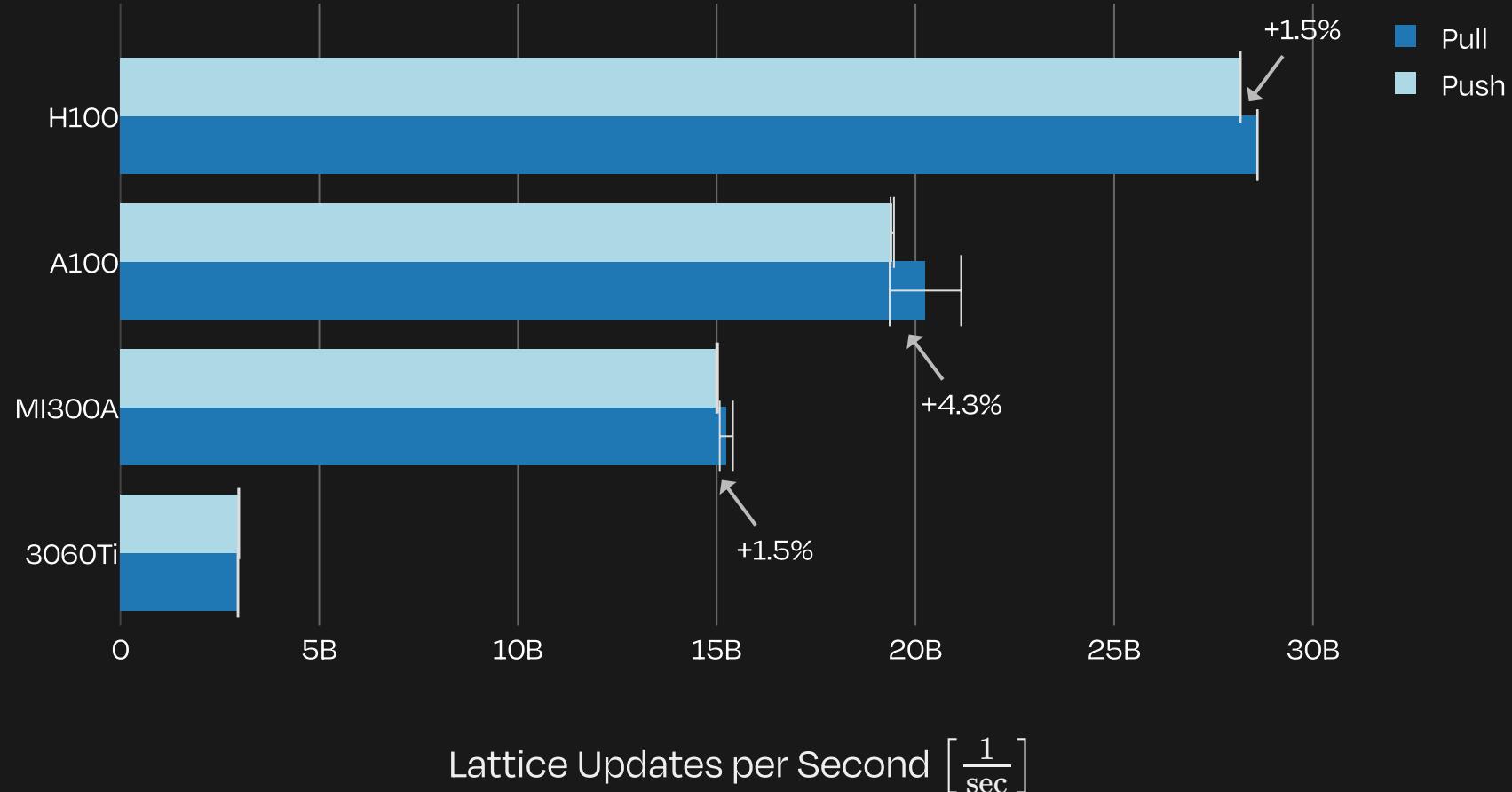
Fused Stream-Collide



Fused Stream-Collide



Results - Push vs. Pull



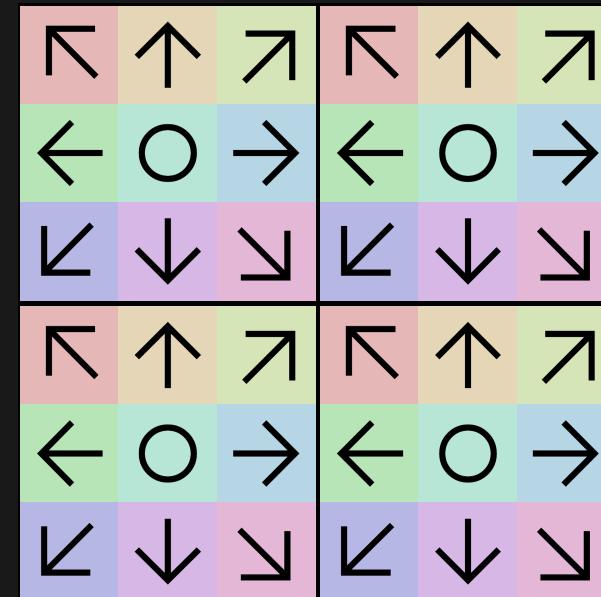
Peak performance: 28.6 BLUPS (H100), 22.0 BLUPS(A100)

A100 and H100 (32000×32000), 3060Ti (3000×3000), 100 steps, ≥ 5 repeats, D2Q9 Shearwave Decay

3. Use Coalescing Memory Accesses

Memory Layout

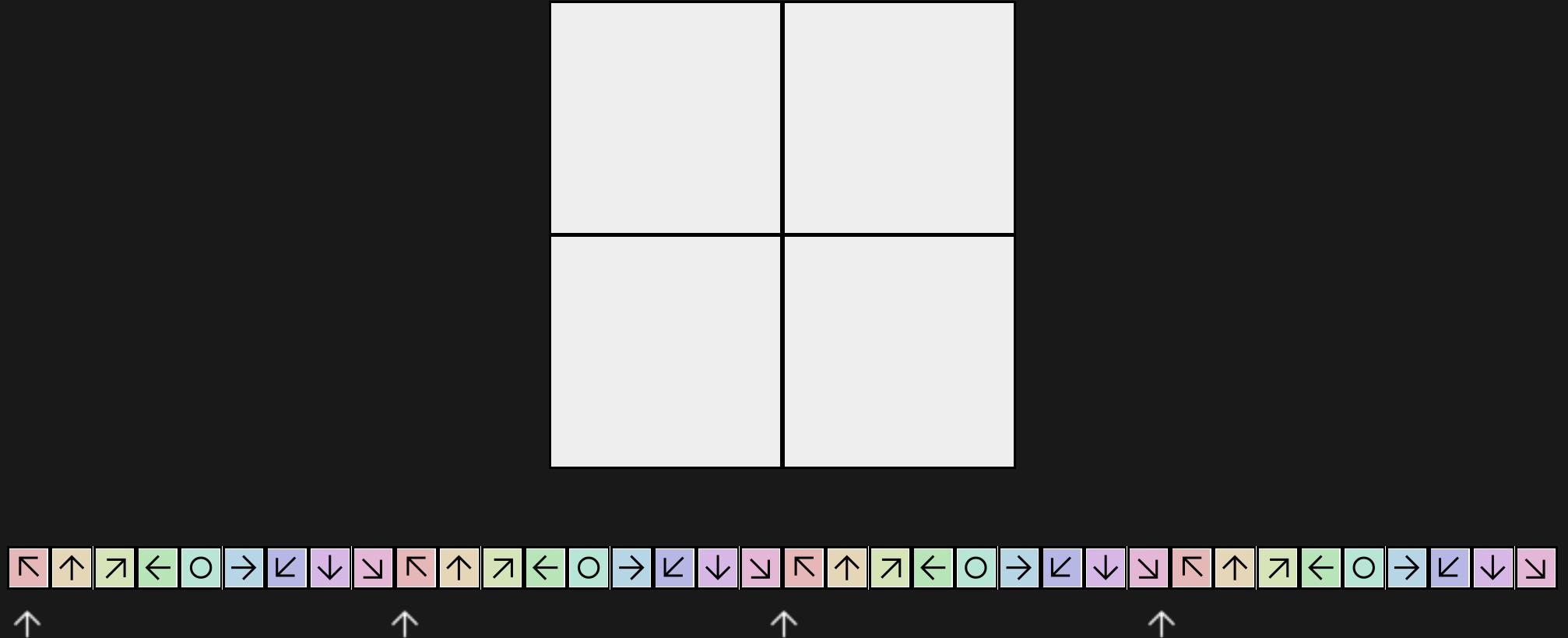
Array of Structs - uncoalesced



[Navarro-Hinojosa et al. 2018, Chapter 3.2]

Memory Layout

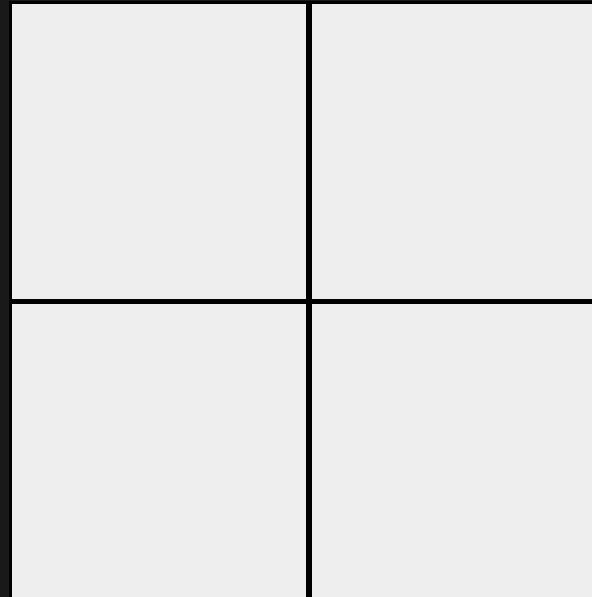
Array of Structs - uncoalesced



[Navarro-Hinojosa et al. 2018, Chapter 3.2]

Memory Layout

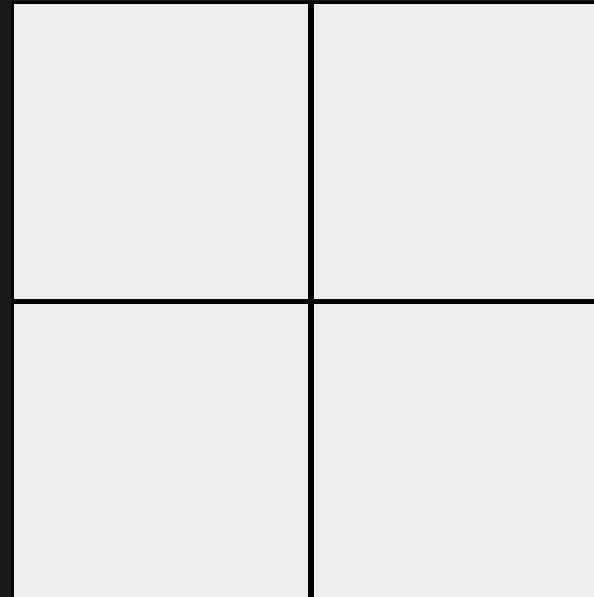
Array of Structs - uncoalesced



[Navarro-Hinojosa et al. 2018, Chapter 3.2]

Memory Layout

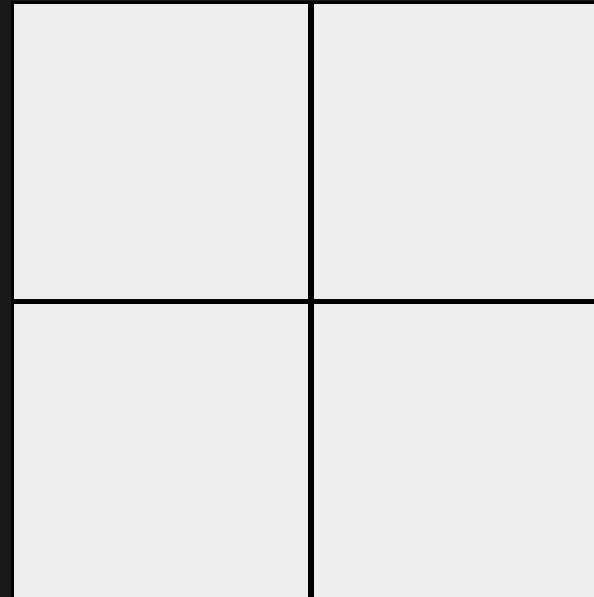
Array of Structs - uncoalesced



[Navarro-Hinojosa et al. 2018, Chapter 3.2]

Memory Layout

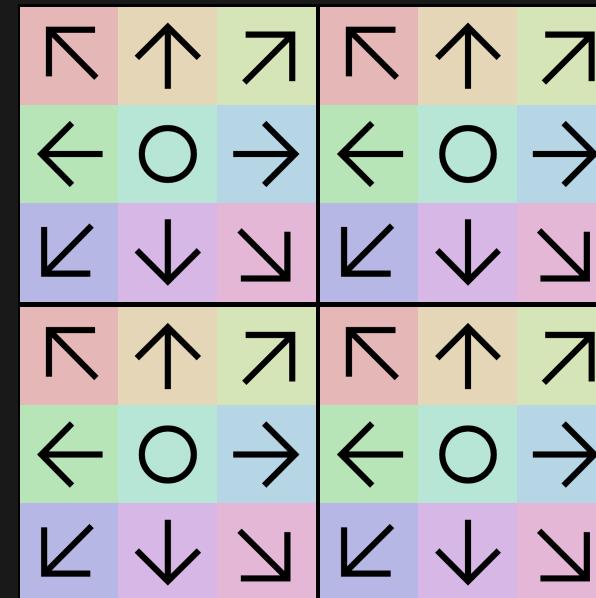
Array of Structs - uncoalesced



[Navarro-Hinojosa et al. 2018, Chapter 3.2]

Memory Layout

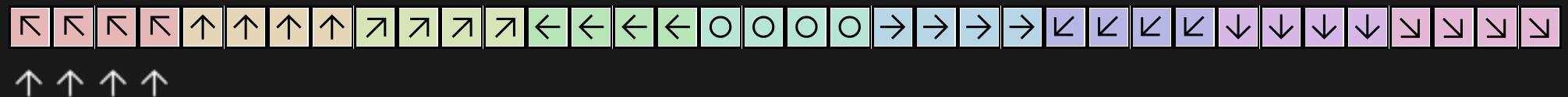
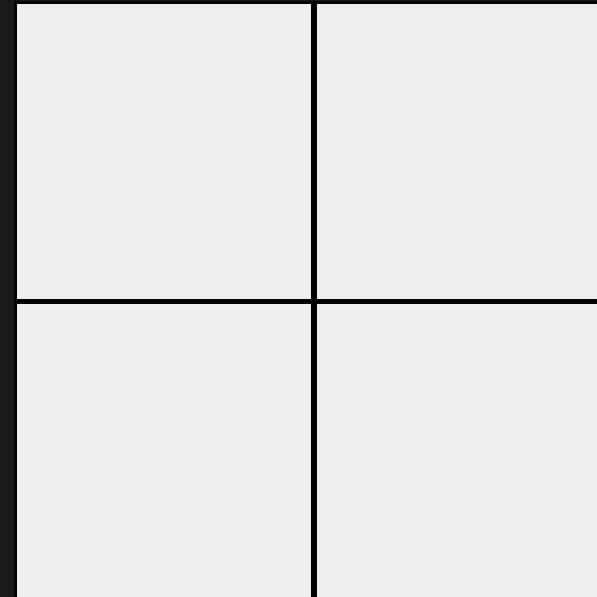
Struct of Arrays - coalesced



[Navarro-Hinojosa et al. 2018, Chapter 3.2]

Memory Layout

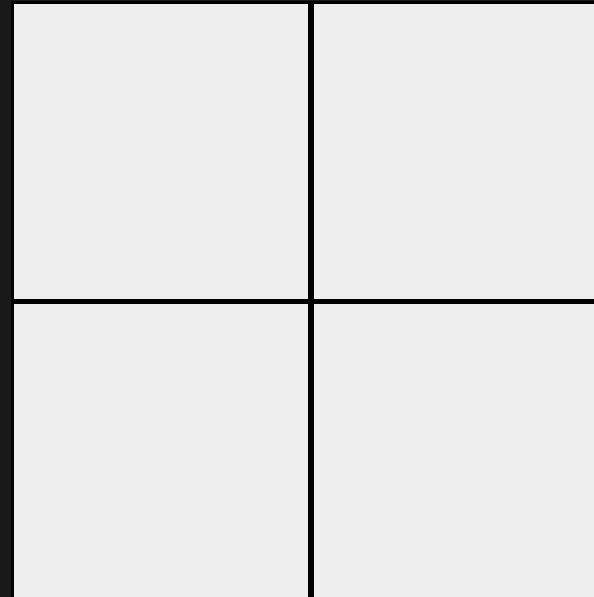
Struct of Arrays - coalesced



[Navarro-Hinojosa et al. 2018, Chapter 3.2]

Memory Layout

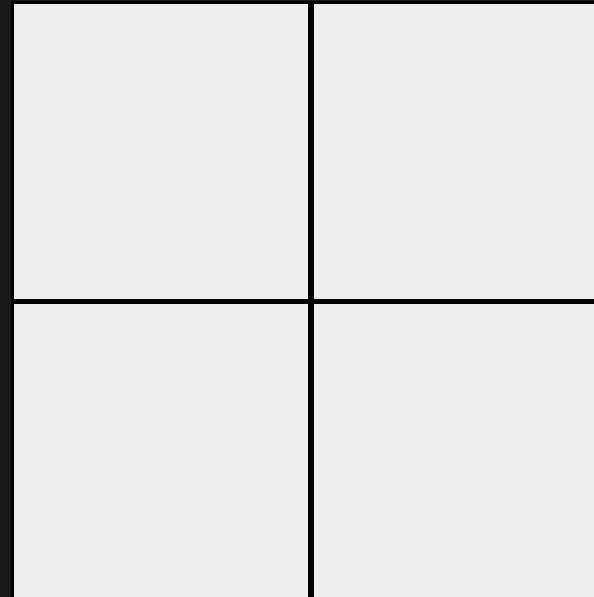
Struct of Arrays - coalesced



[Navarro-Hinojosa et al. 2018, Chapter 3.2]

Memory Layout

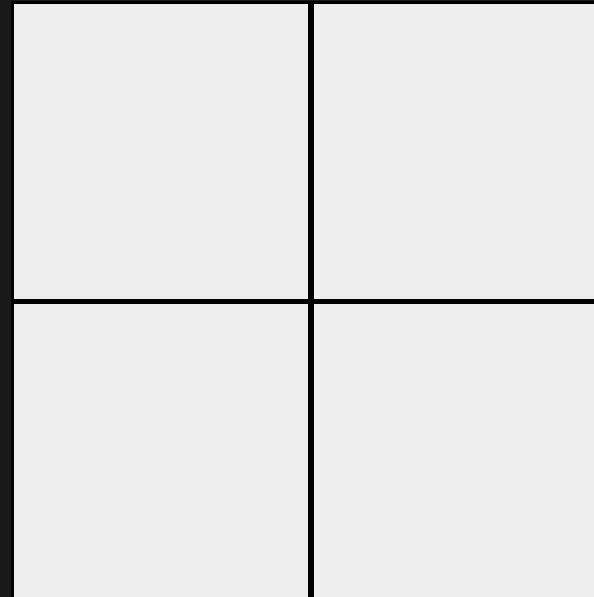
Struct of Arrays - coalesced



[Navarro-Hinojosa et al. 2018, Chapter 3.2]

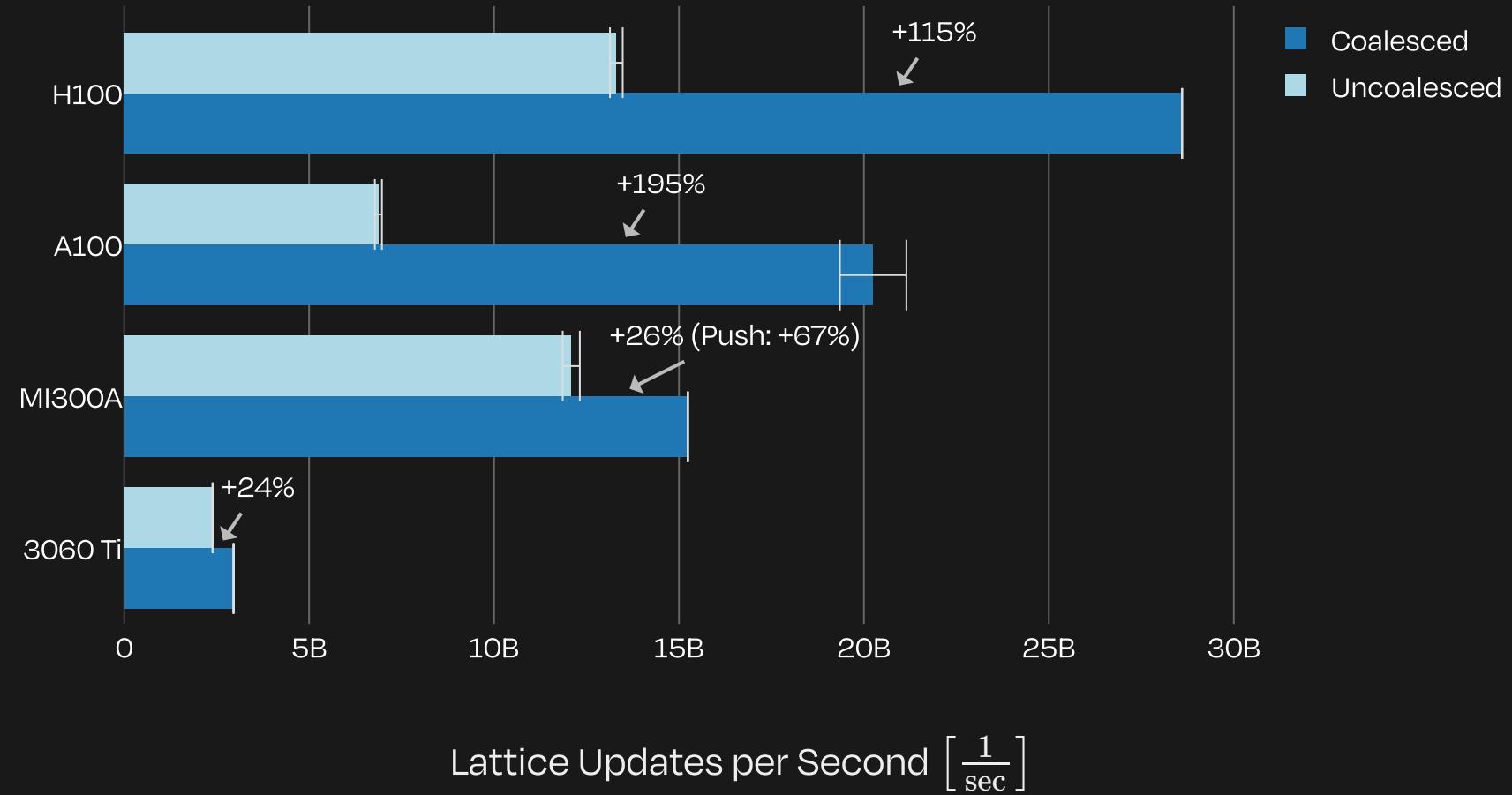
Memory Layout

Struct of Arrays - coalesced



[Navarro-Hinojosa et al. 2018, Chapter 3.2]

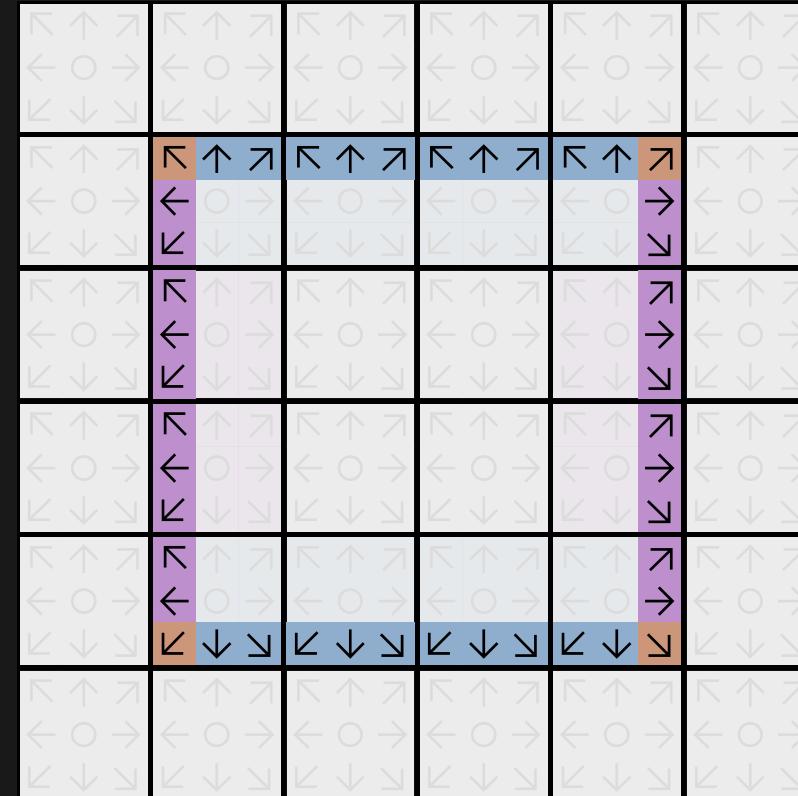
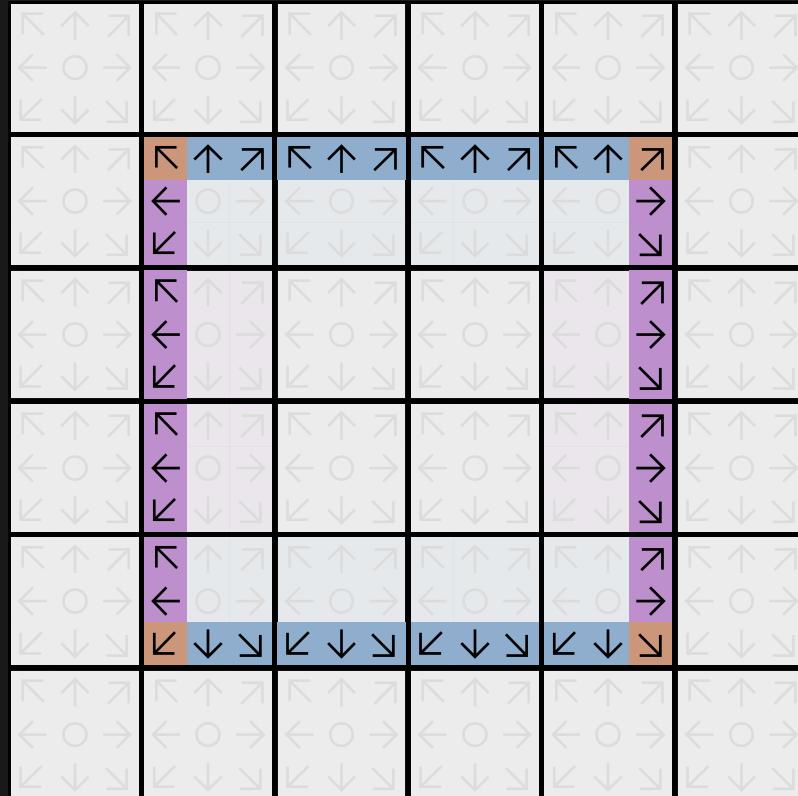
Results - Coalescing Accesses



Pull scheme, A100 and H100 (32000×32000), 3060Ti (3000×3000), 100 steps, ≥ 5 repeats, D2Q9 Shearwave Decay

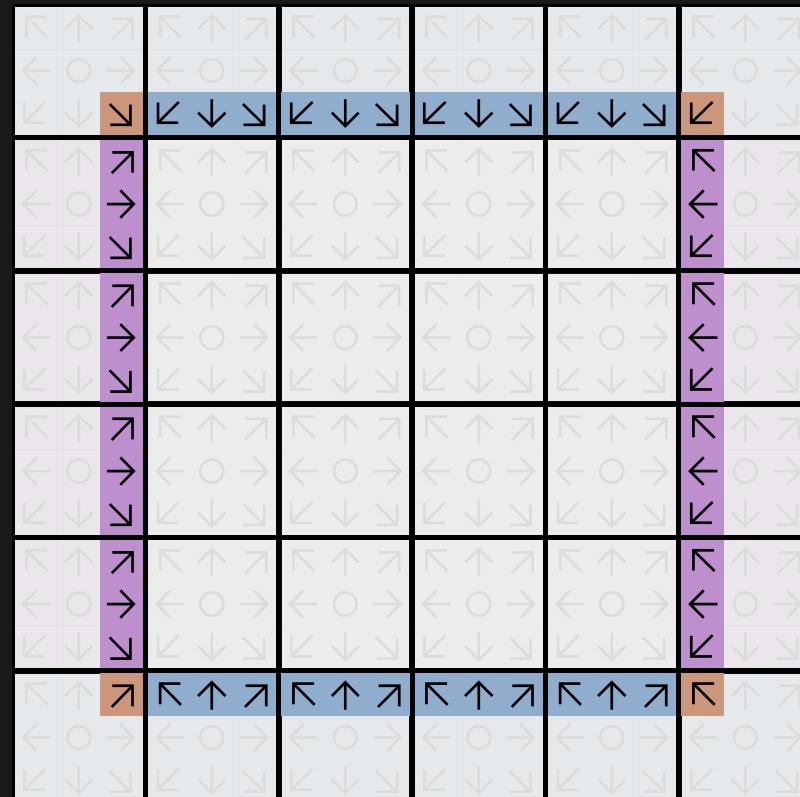
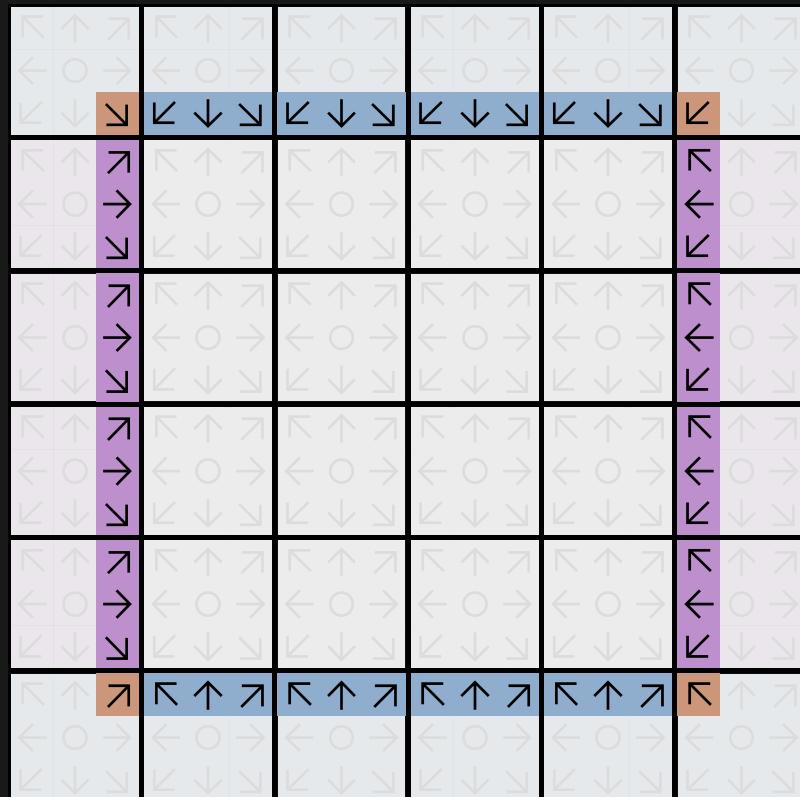
4. Work while communicating

MPI Communication



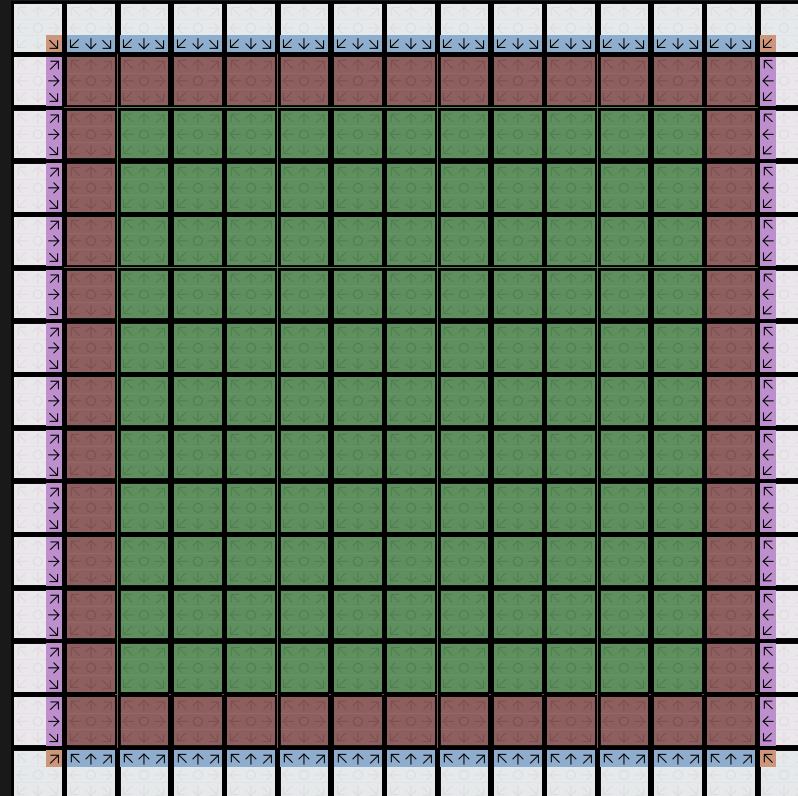
Communicate each border and corner asynchronously

MPI Communication



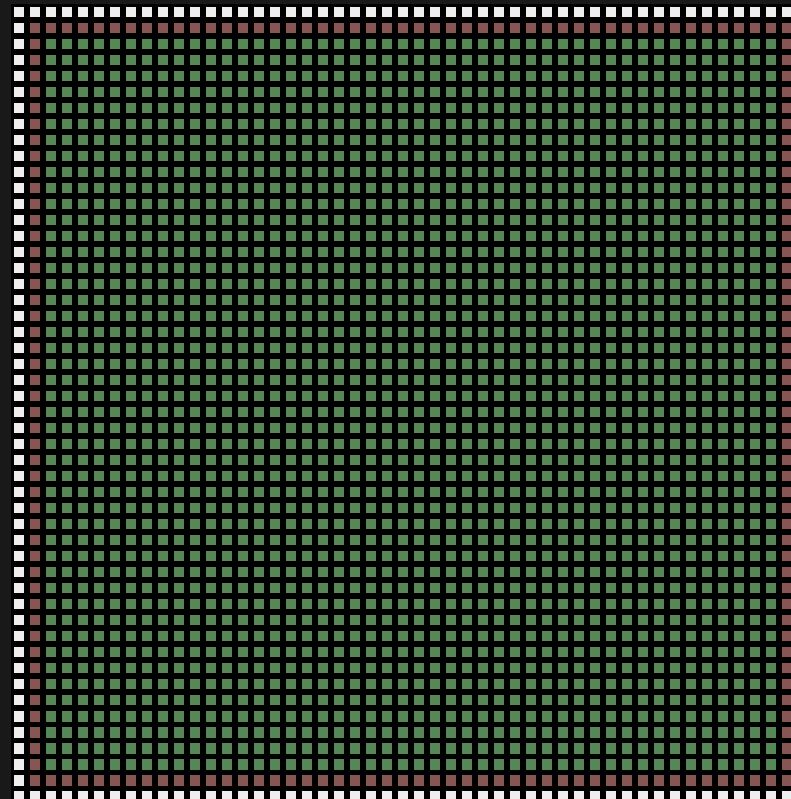
Communicate each border and corner asynchronously

MPI Communication



$\mathcal{O}(N)$ halo nodes $\ll \mathcal{O}(N^2)$ inner nodes

MPI Communication



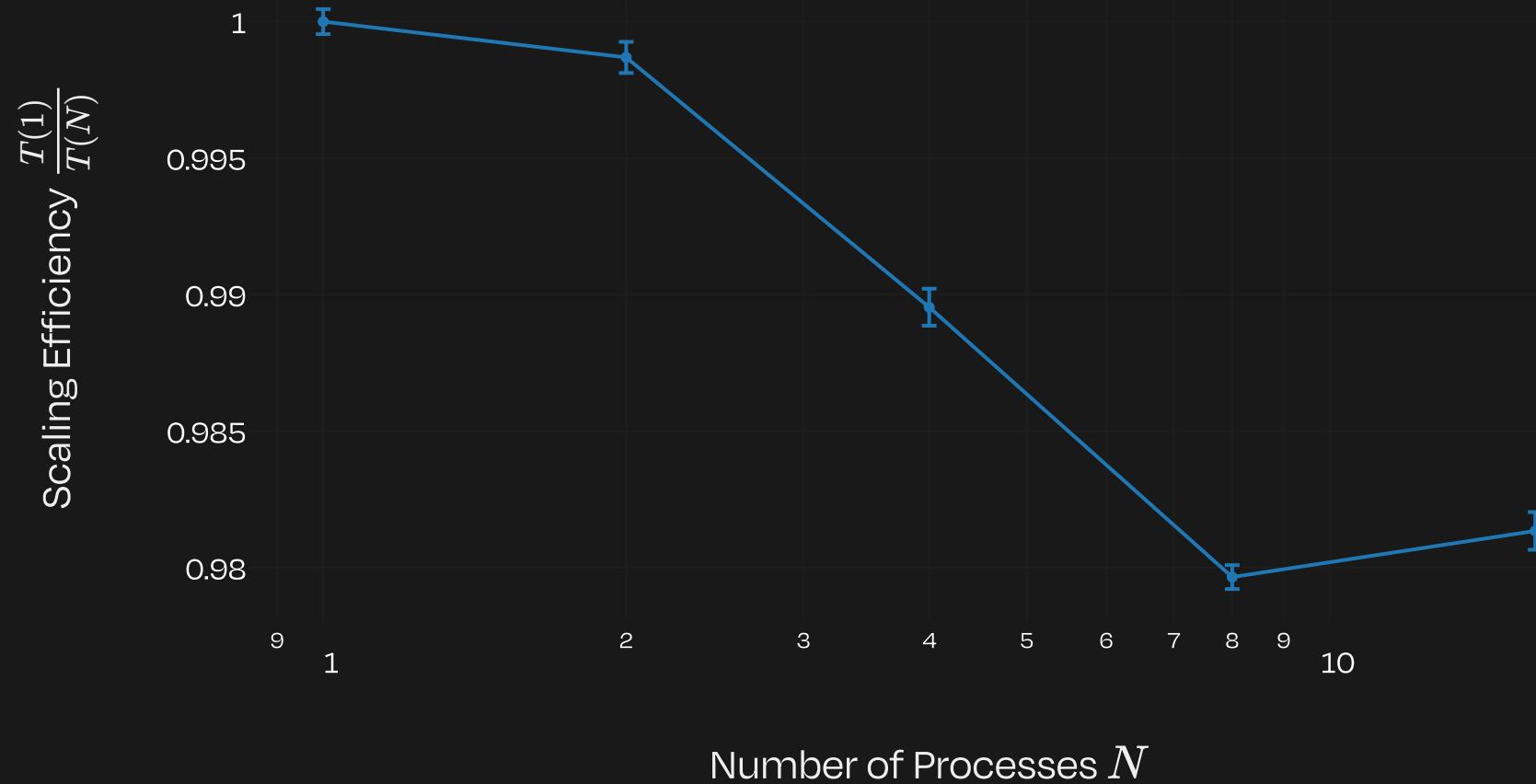
$\mathcal{O}(N)$ halo nodes $\ll \mathcal{O}(N^2)$ inner nodes

MPI Communication

1. Post all `IRecv`
2. Pack Buffers and `ISend` them
3. Work on inner nodes during data transfer!
4. Block only if transfer not yet completed
5. Work on outer nodes¹

1. The Kokkos Lectures Module 6, Sandia National Laboratories, 2020

Results - Weak Scaling



Nvidia A100 GPUs, 32000×32000 nodes per process, 100 steps, 5 repeats, D2Q9 Shearwave Decay

Thank you for your attention!

Bonus: Taichi



Taichi Features

- Embedded in Python
- Portable, fast, statically typed
- Layout- and Dimension-independent Code
- GUI, LAS, Solvers etc. included
- Differentiable
- Only $\sim 15\%$ performance penalty¹

¹. 10^4 steps, 1000^2 nodes of D2Q9-PBC on 3060Ti compared to peak of Kokkos (2503 MLUPS vs. 2956 MLUPS)

Full example

```
1 # Imports
2 import taichi as ti
3 import taichi.math as tm
4
5 # Start Taichi
6 ti.init(arch=ti.gpu)
7
8 # Define constants
9 NY = 800
10 NX = 800
11 Q = 9
12 OMEGA = 1.7
13 RHO = 1.
14 U_0 = 0.1
15 w = [4/9, 1/9, 1/9, 1/9, 1/9, 1/36, 1/36, 1/36, 1/36]
16 cx = [0,1,0,-1,0,1,-1,-1,1]
17 cy = [0,0,1,0,-1,1,1,-1,-1]
18 refl = [0,3,4,1,2,7,8,5,6]
19
20 # Define fields
21 buf = ti.field(dtype=ti.f32, shape=(NX,NY,Q,))
22 pixels = ti.Vector.field(n=3, dtype=ti.f32, shape=(NX,NY))
```

≤ 50 Lines of Code, including GUI

Full example

