AQTIVATE Conference, Larnaca, 1.10.2026



Funded by
the European Union









High-performance simulations of higher representations of Wilson fermions

Sofie Martins, University of Graz

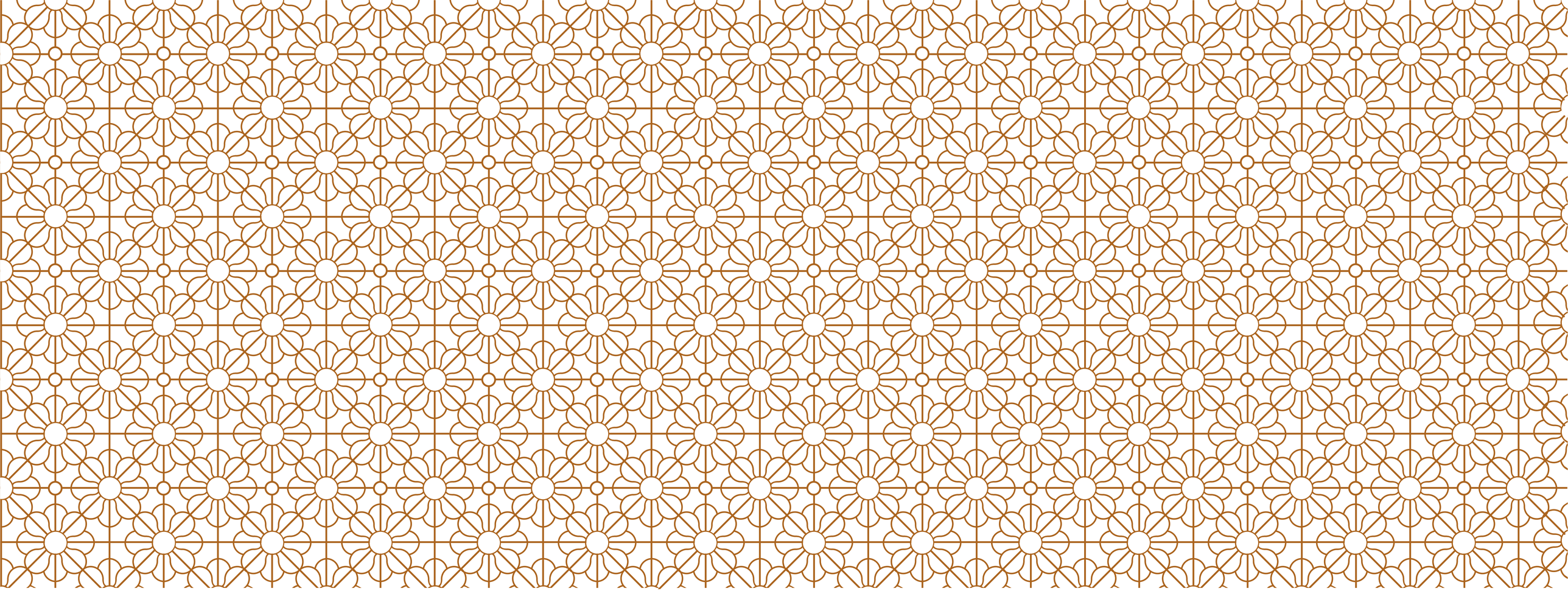
In collaboration with Vincent Drach, Claudio Pica and Antonio Rago

This project received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No.813942 (EuroPLEx)

Based on Drach et. al, Comput. Phys. Commun., 322 (2026) 110061

Outline

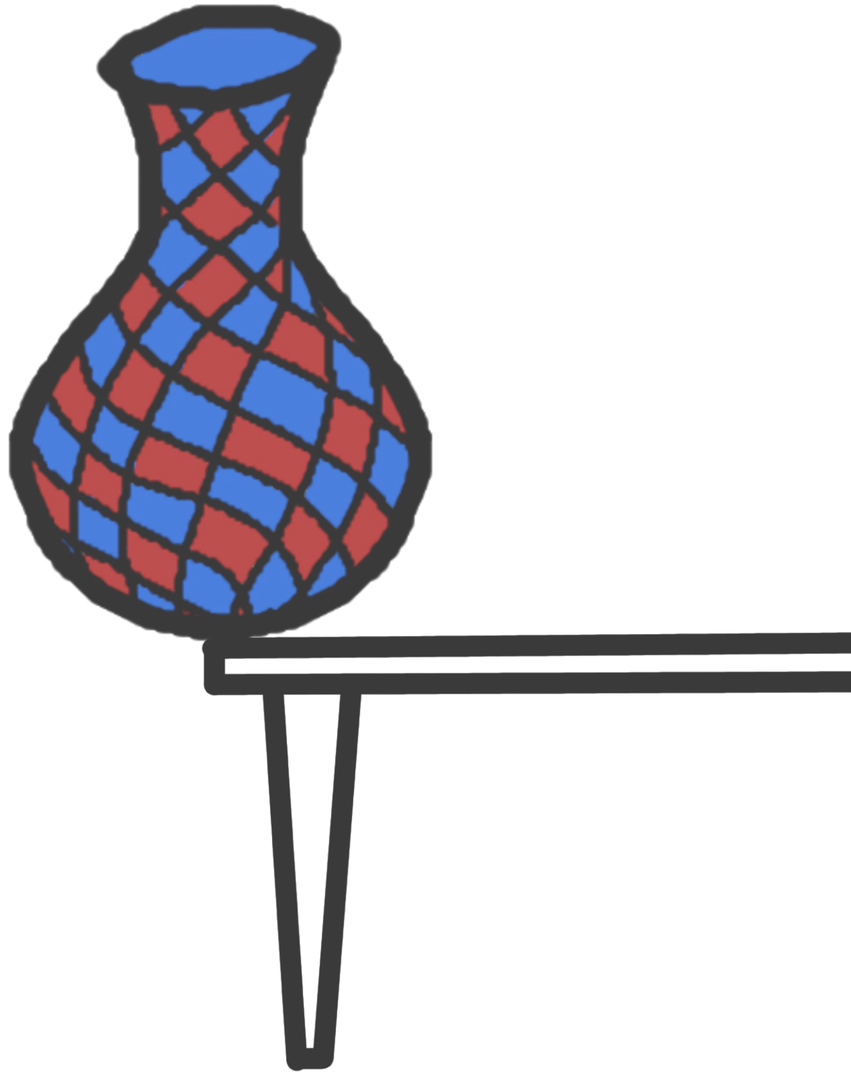
- Technicolor/Composite Higgs
- Optimization of a lattice code that can simulate CH/TC theories



COMPOSITE HIGGS THEORIES

WHY COMPOSITE HIGGS?

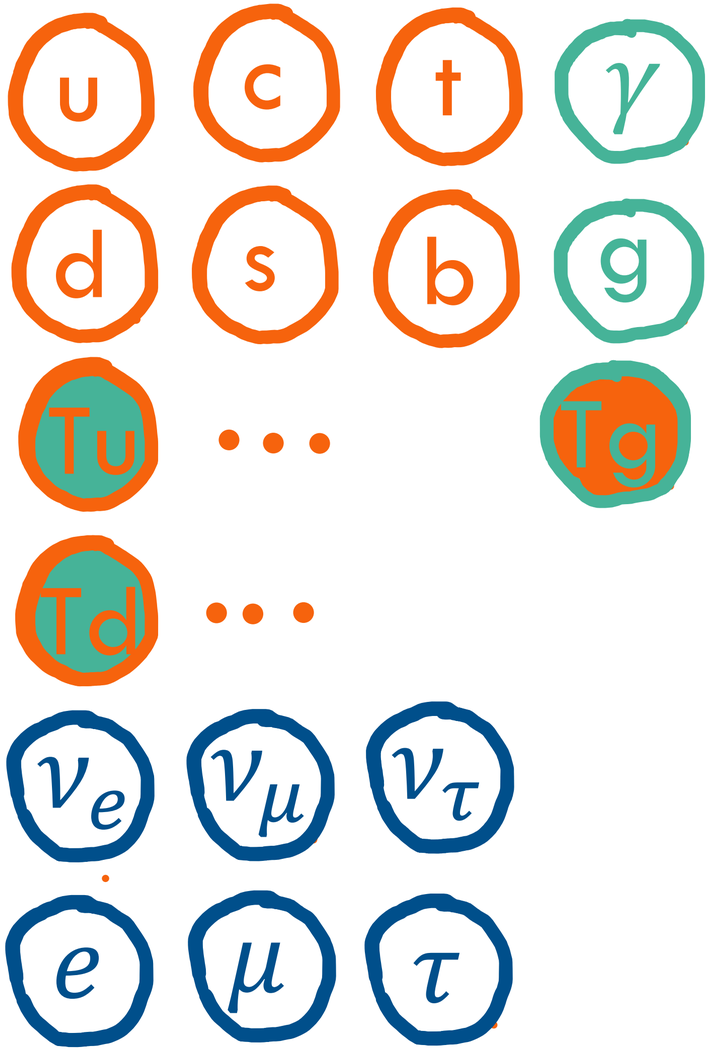




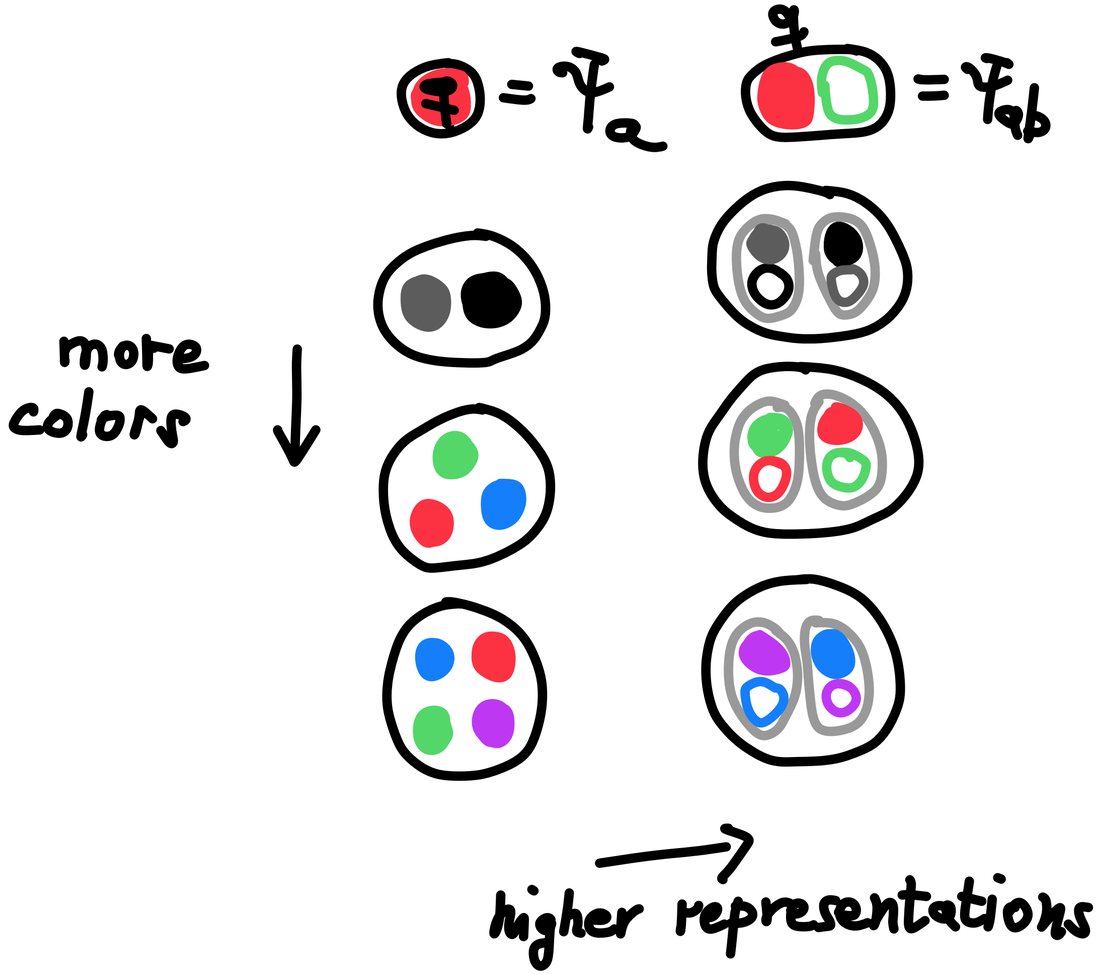
WHY COMPOSITE HIGGS?

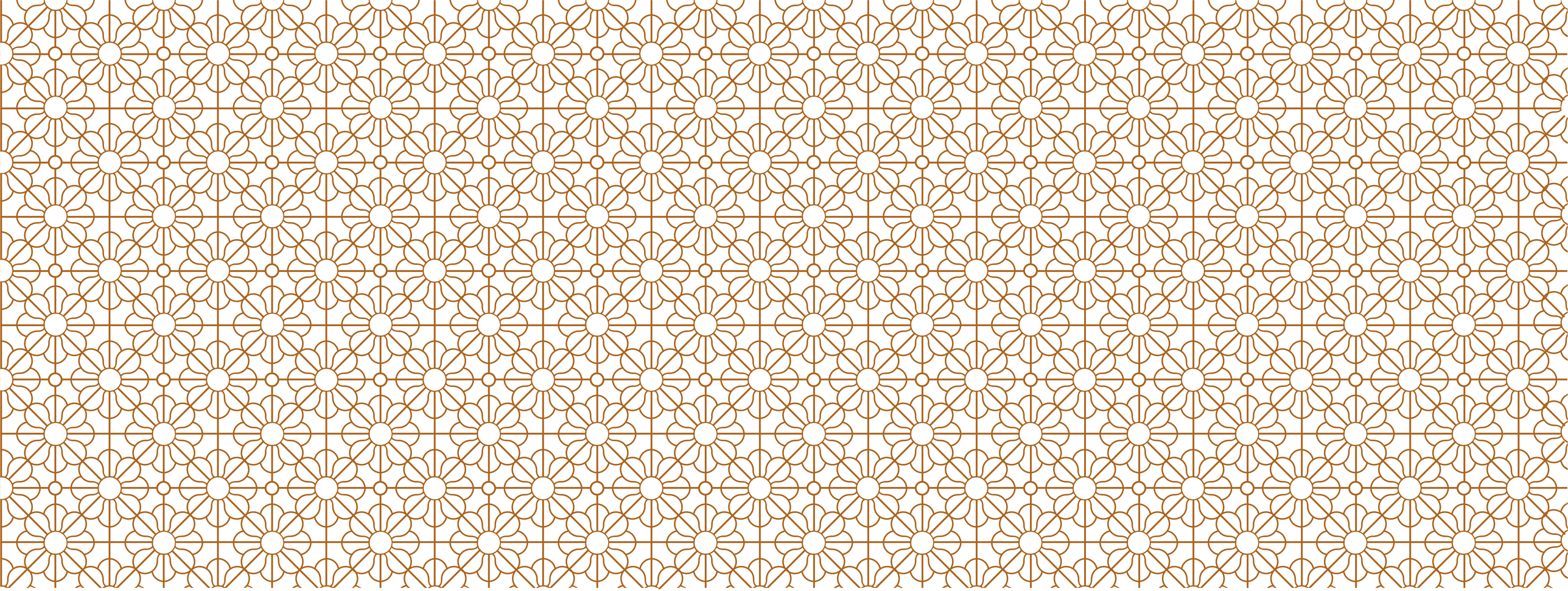
Replace weak + custodial with new sector

$$SU(3)_c \times \boldsymbol{SU(2)_L} \times \boldsymbol{SU(2)_R} \times U(1)_Y$$
$$\rightarrow$$
$$SU(3)_c \times U(1)_Y \times SU(N_{FC})$$



WHY COMPOSITE HIGGS?

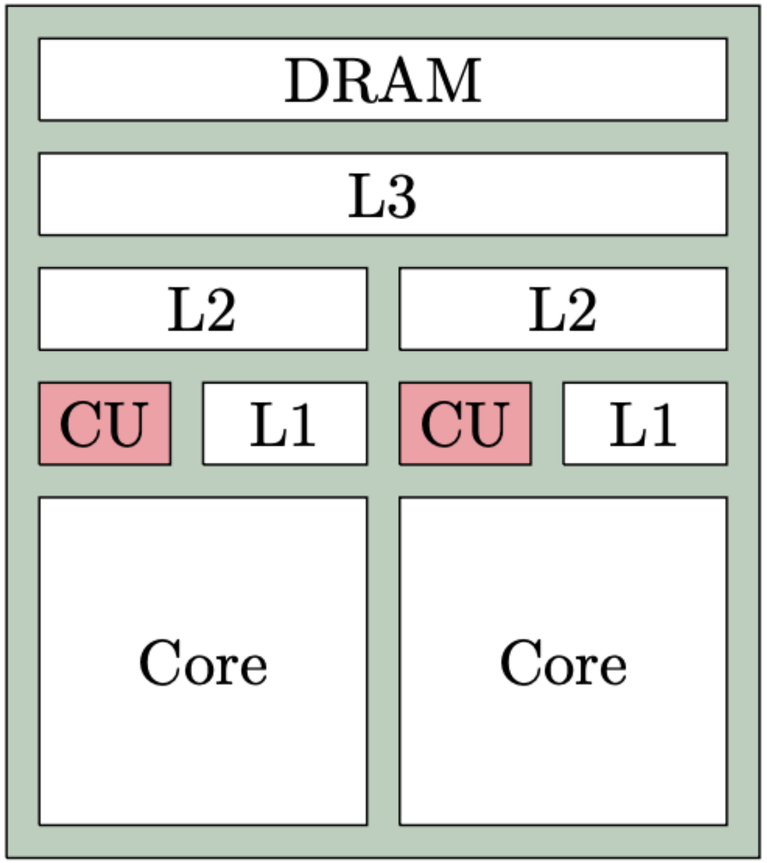


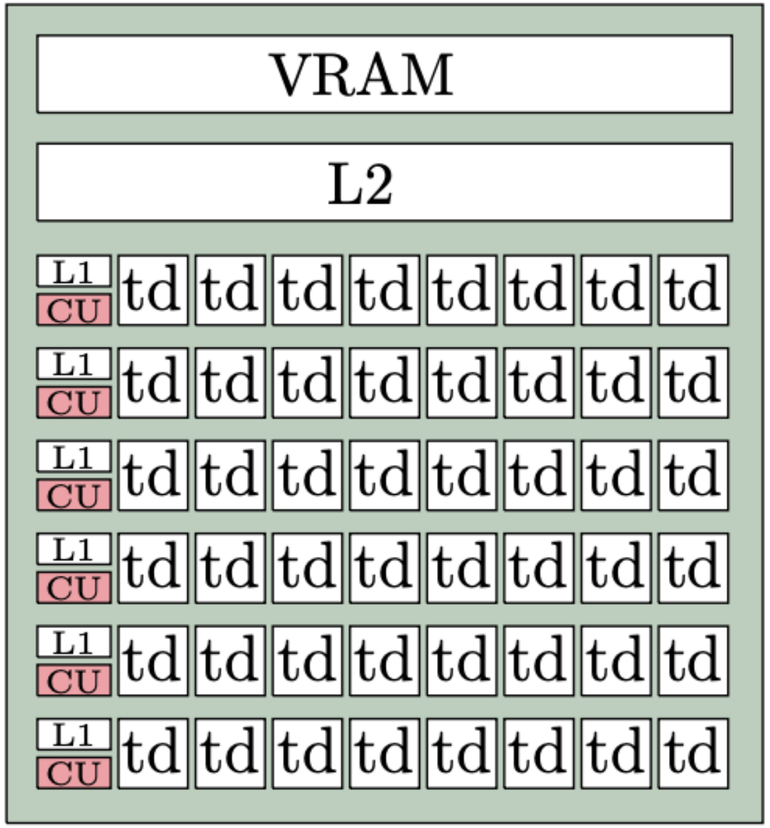


HIGH-PERFORMANCE COMPUTING (HPC)

For lattice computations

CPUS VS. GPUS





THE BOTTLENECK IS THE DIRAC OPERATOR



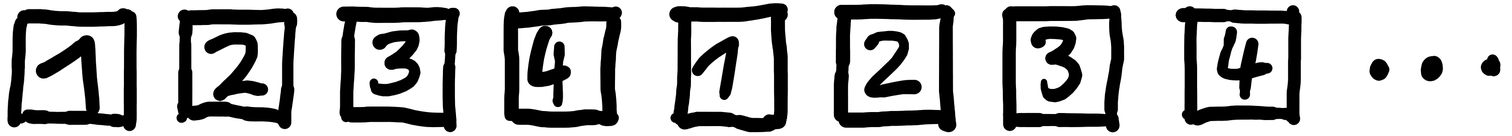
The programmer needs to know everything about the hardware and experiment with the code

- Memory: Global, shared, constant, registers
- Parallelisation strategy
- Block and grid dimensions
- Bandwidth and compute throughput

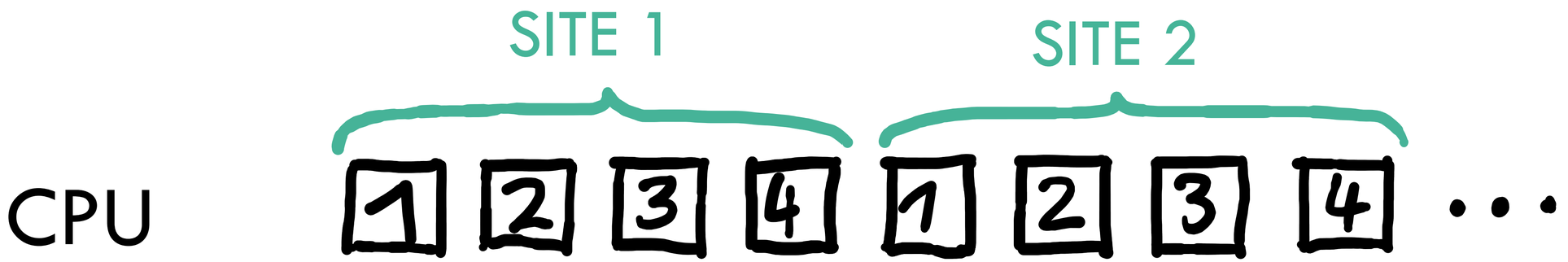
OPTIMIZING THE DIRAC OPERATOR
COALESCED MEMORY ACCESS



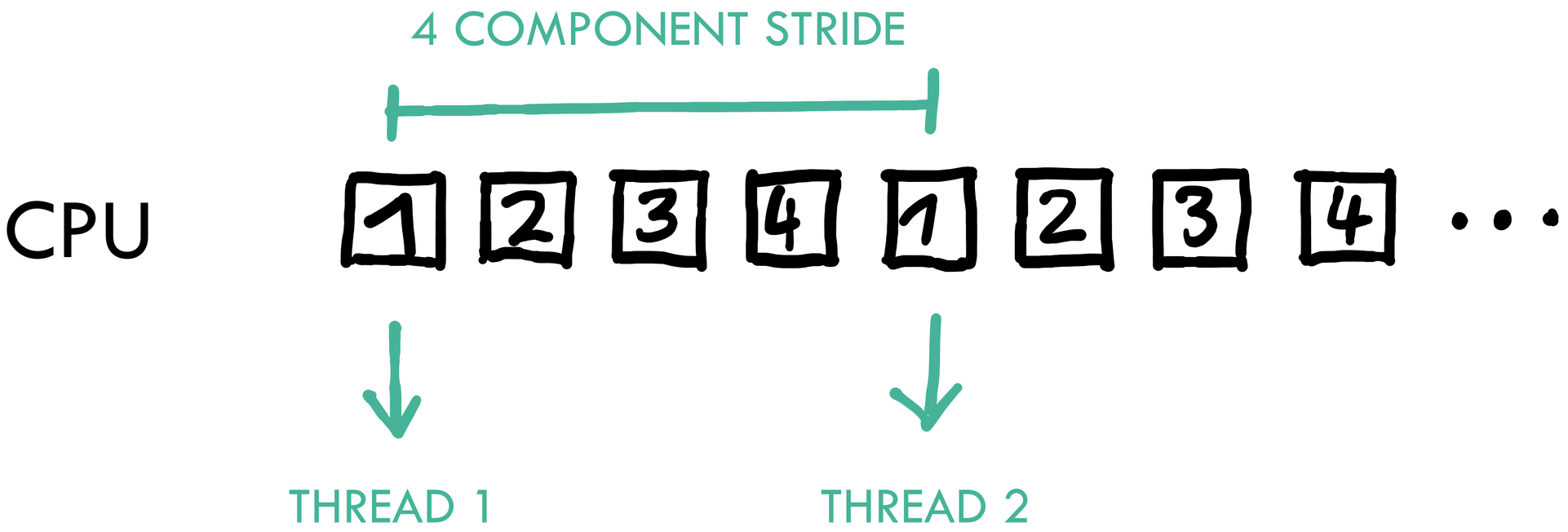
CPU



OPTIMIZING THE DIRAC OPERATOR
COALESCED MEMORY ACCESS



OPTIMIZING THE DIRAC OPERATOR
COALESCED MEMORY ACCESS



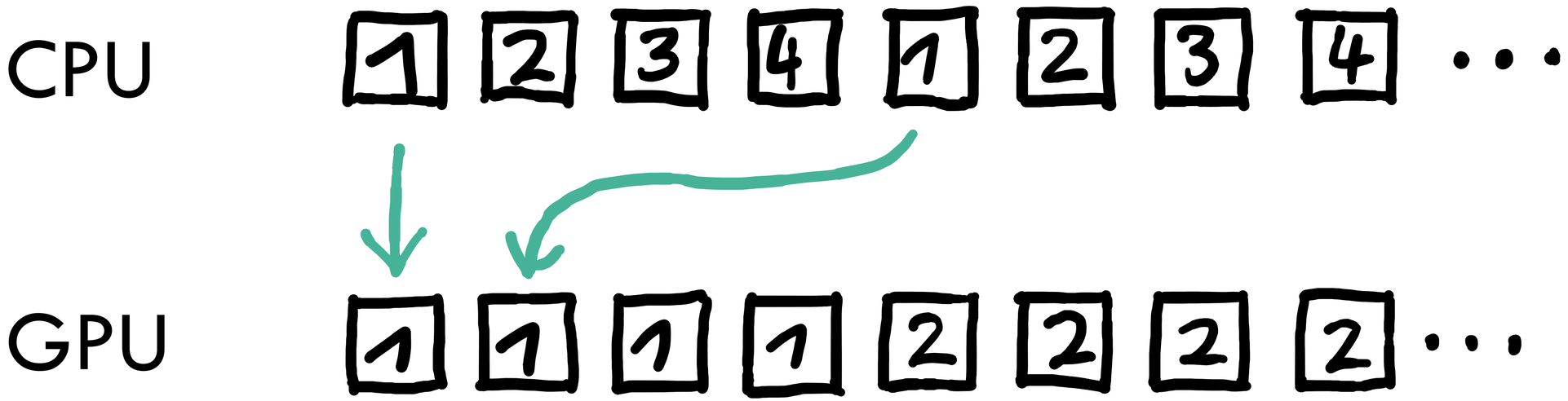
OPTIMIZING THE DIRAC OPERATOR
COALESCED MEMORY ACCESS



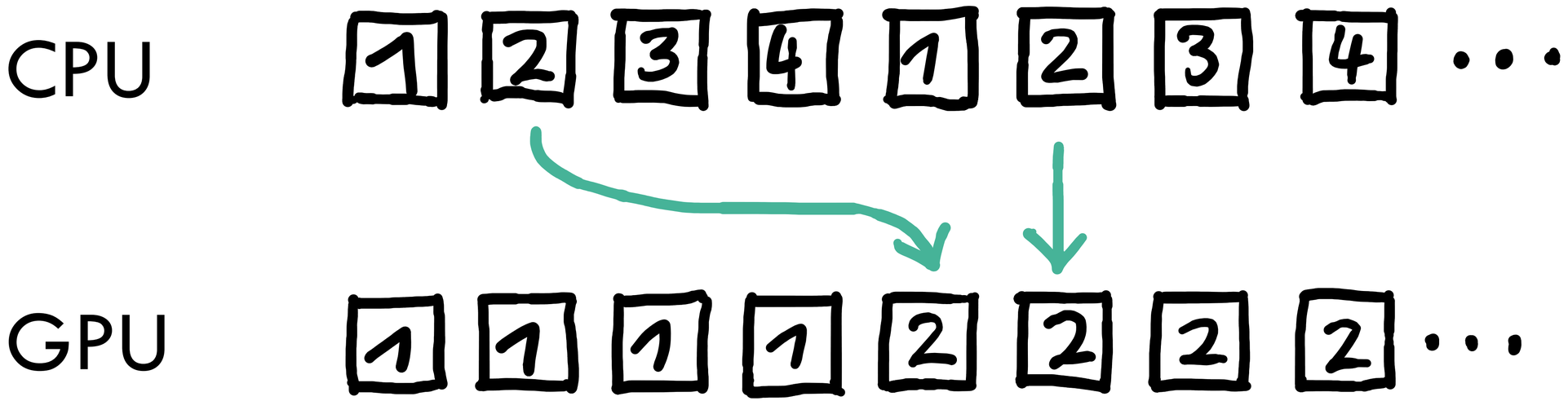




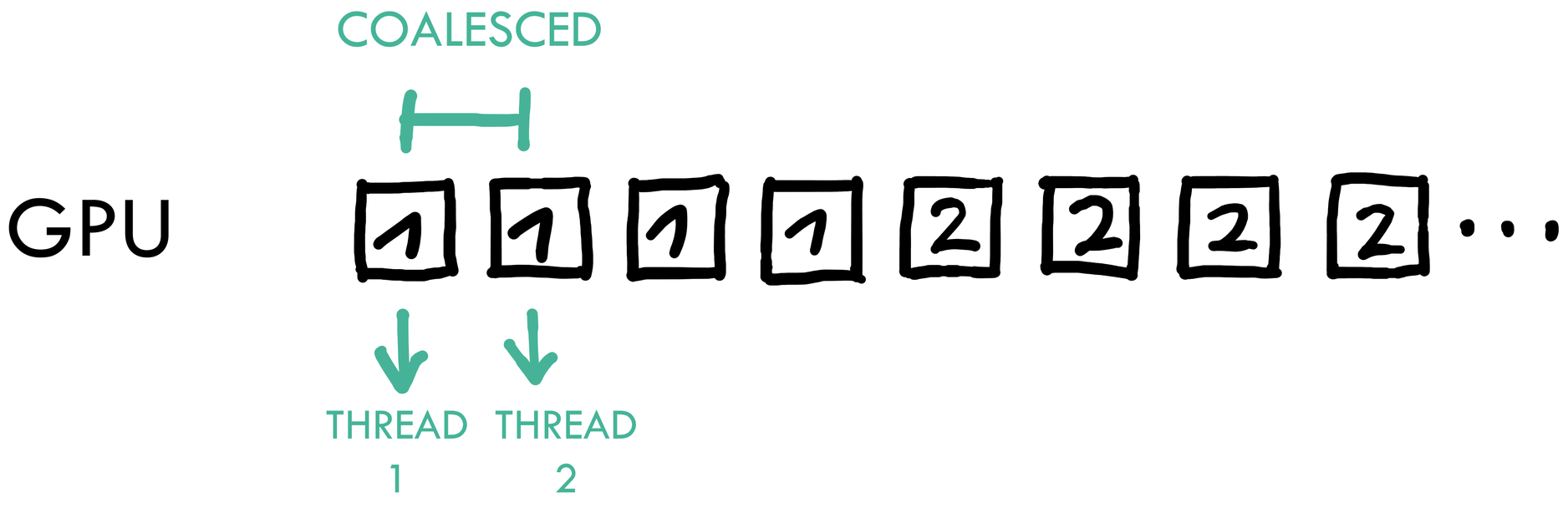
OPTIMIZING THE DIRAC OPERATOR
COALESCED MEMORY ACCESS



OPTIMIZING THE DIRAC OPERATOR
COALESCED MEMORY ACCESS



OPTIMIZING THE DIRAC OPERATOR
COALESCED MEMORY ACCESS

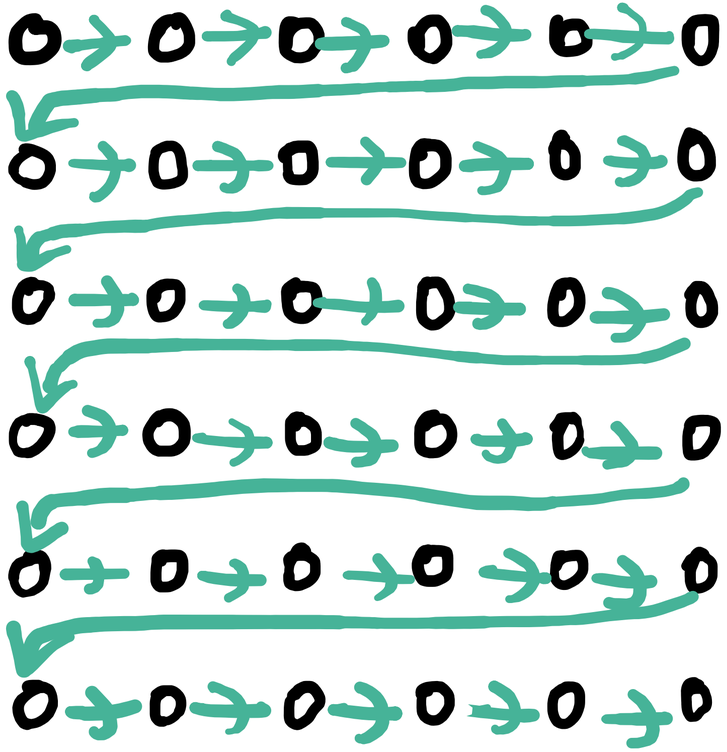


MAXIMIZING L2 CACHE HIT RATE

OPTIMIZING THE DIRAC OPERATOR
L2 CACHE OPTIMIZATION



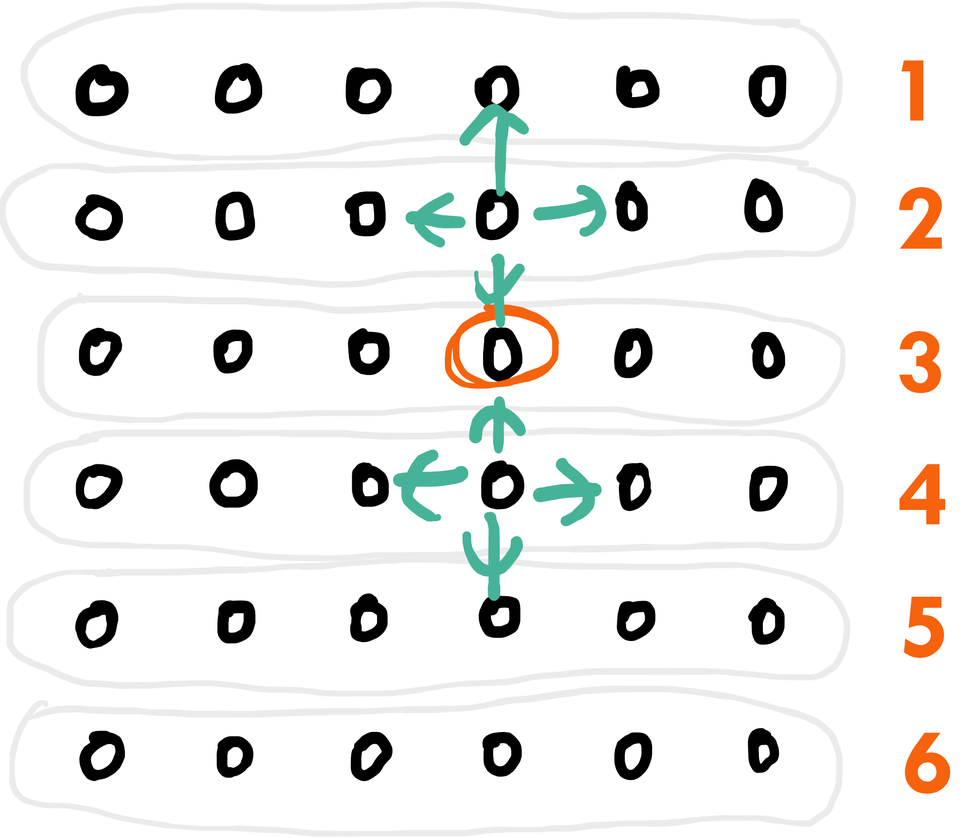
LEXICOGRAPHIC INDEXING



OPTIMIZING THE DIRAC OPERATOR
L2 CACHE OPTIMIZATION

REUSE:
LOW CACHE LOCALITY

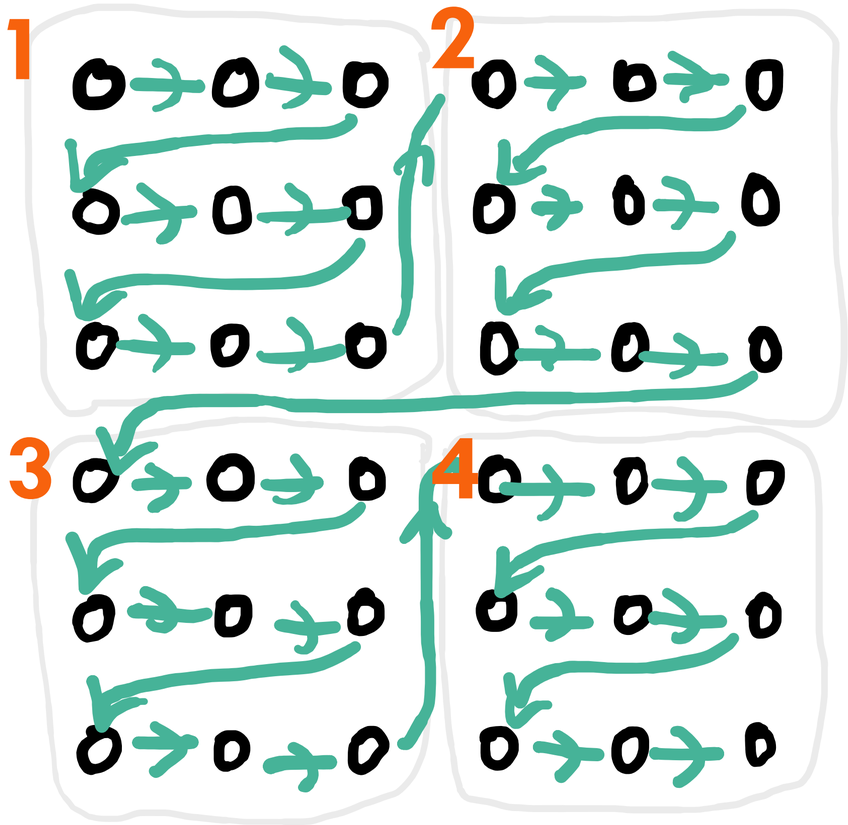
Two sites that share a neighbor are not in closely located threads



OPTIMIZING THE DIRAC OPERATOR
L2 CACHE OPTIMIZATION

SUBBLOCKING

A strategy to solve this is to arrange the sites in consecutive blocks

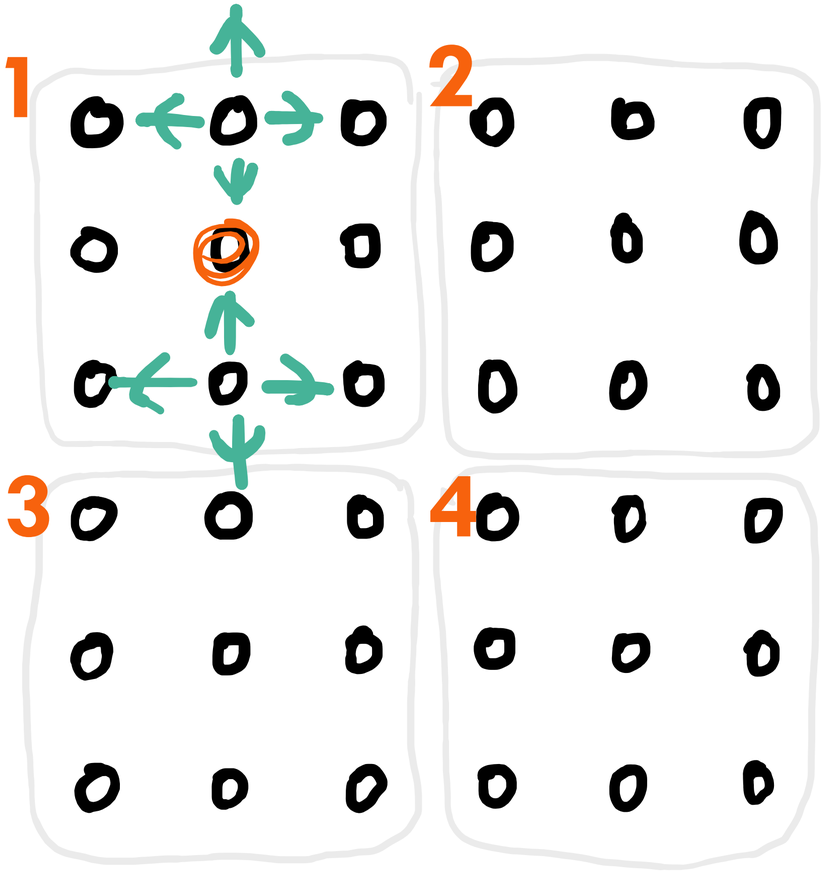


OPTIMIZING THE DIRAC OPERATOR
L2 CACHE OPTIMIZATION



SUBBLOCKING:
HIGH CACHE LOCALITY

Two sites that share a neighbor are in closely located threads

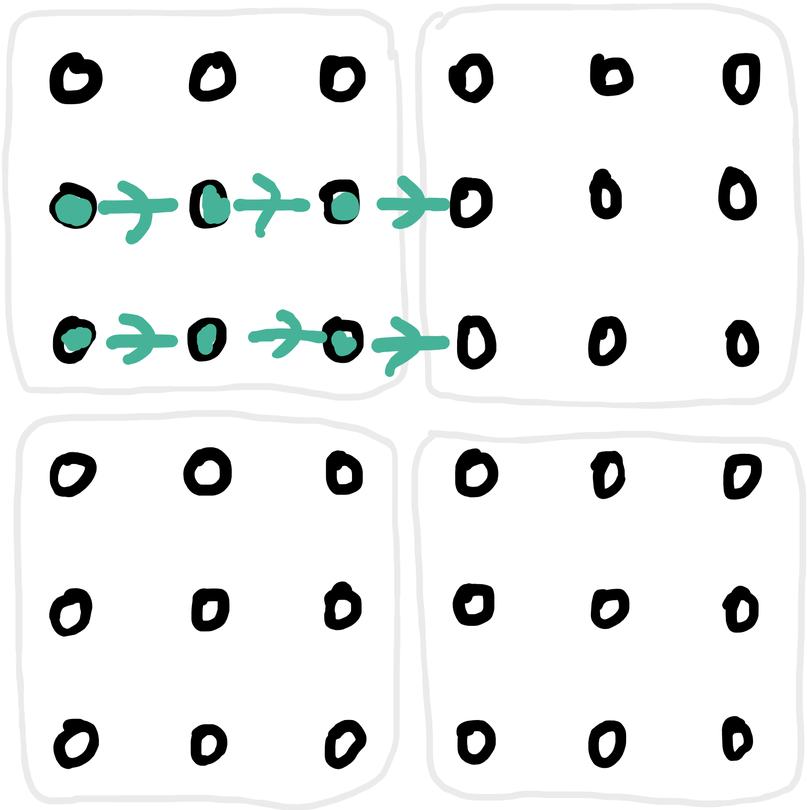


COALESCING DISCRETE DIFFERENTIAL OPERATORS

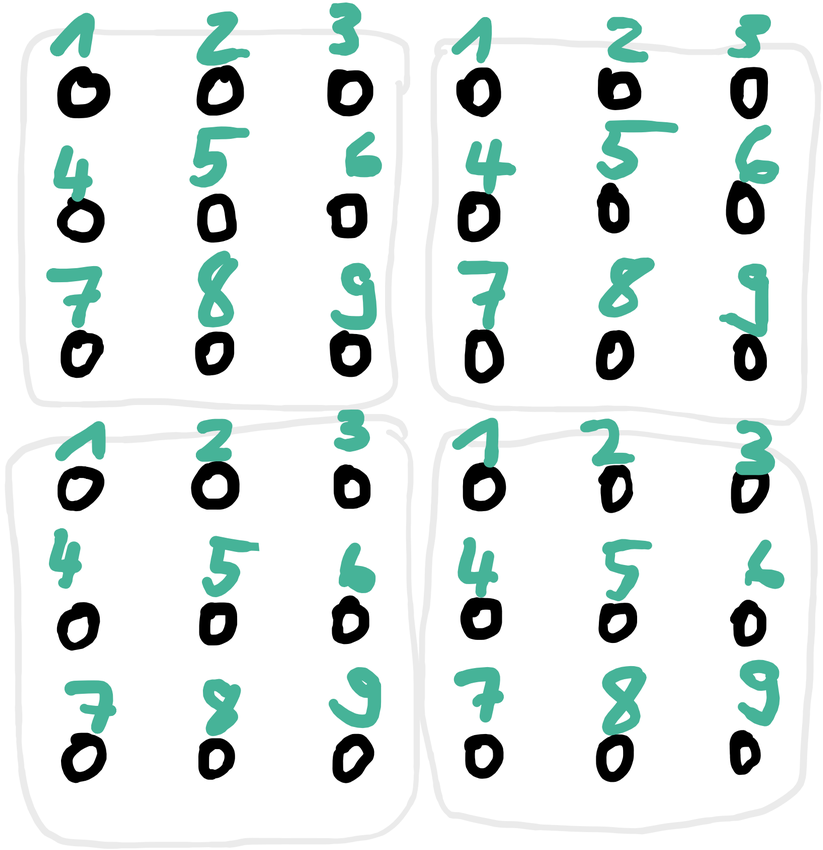
OPTIMIZING THE DIRAC OPERATOR FULLY COALESCED NEIGHBORS?



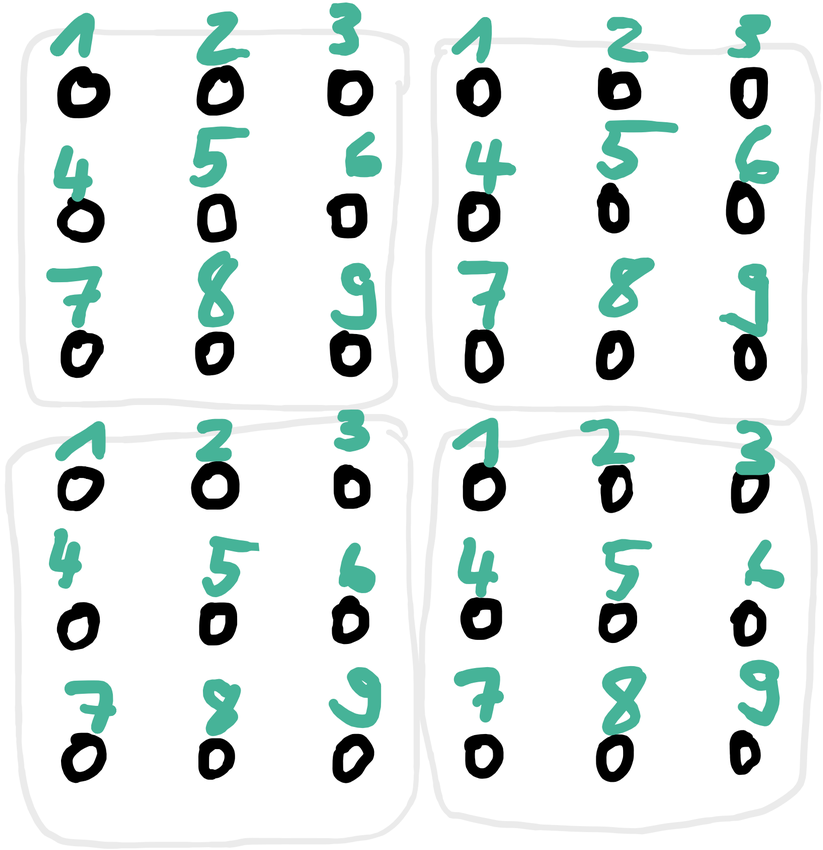
The central sites are coalesced



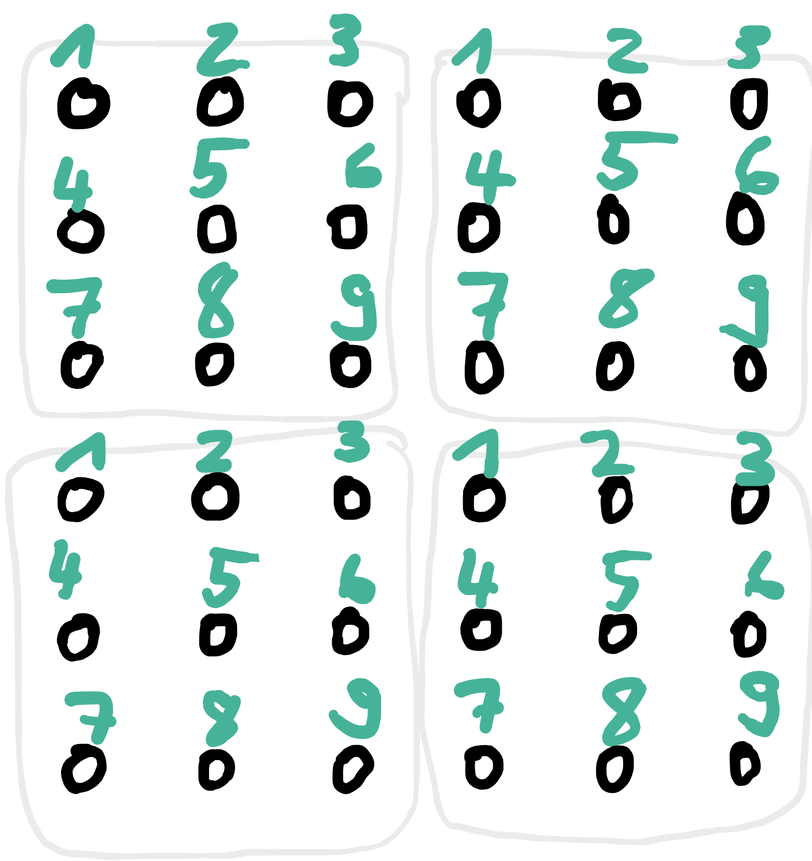
OPTIMIZING THE DIRAC OPERATOR
FULLY COALESCED NEIGHBORS?

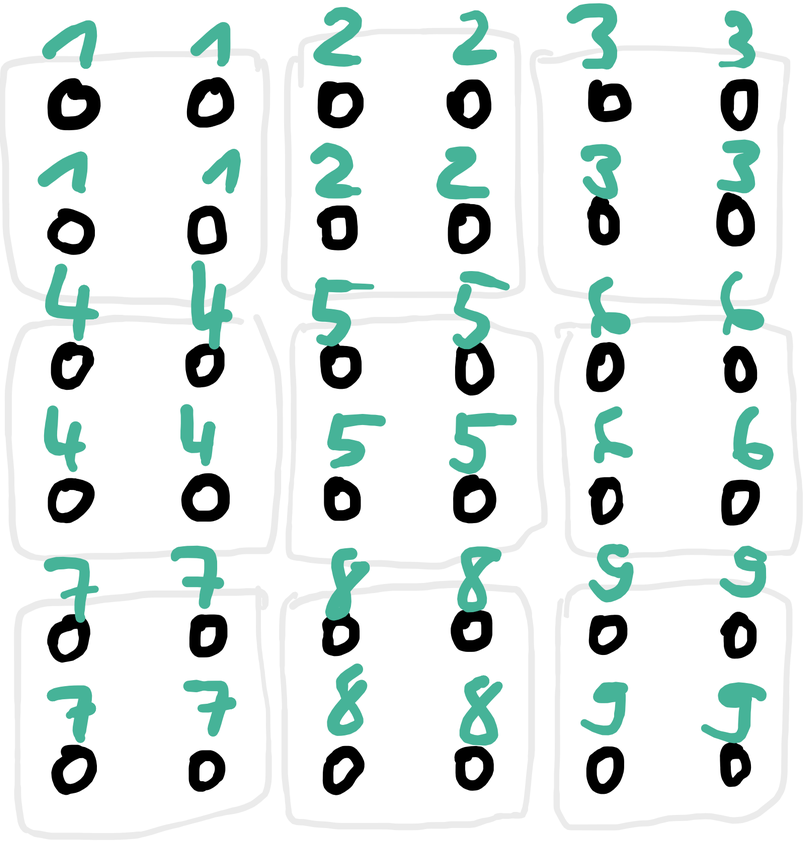


OPTIMIZING THE DIRAC OPERATOR
FULLY COALESCED NEIGHBORS?

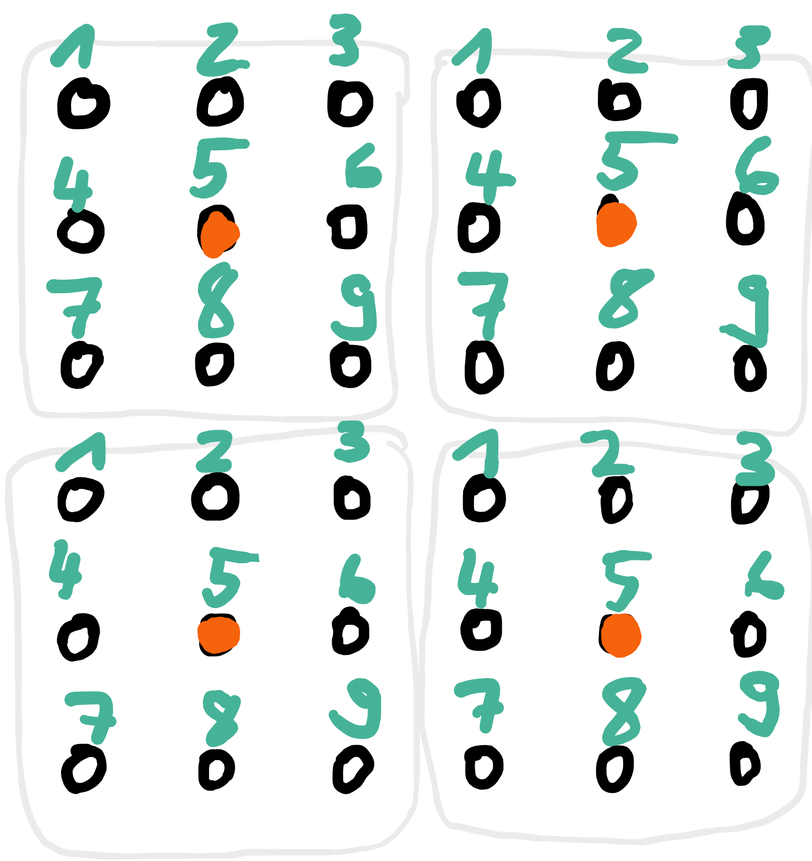


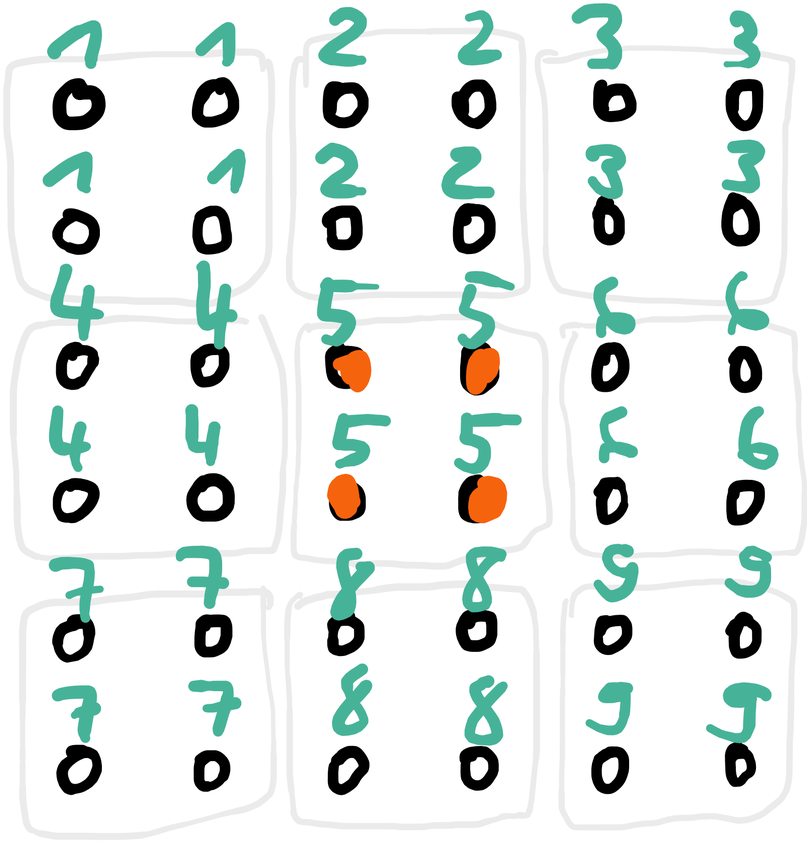
OPTIMIZING THE DIRAC OPERATOR
FULLY COALESCED NEIGHBORS?





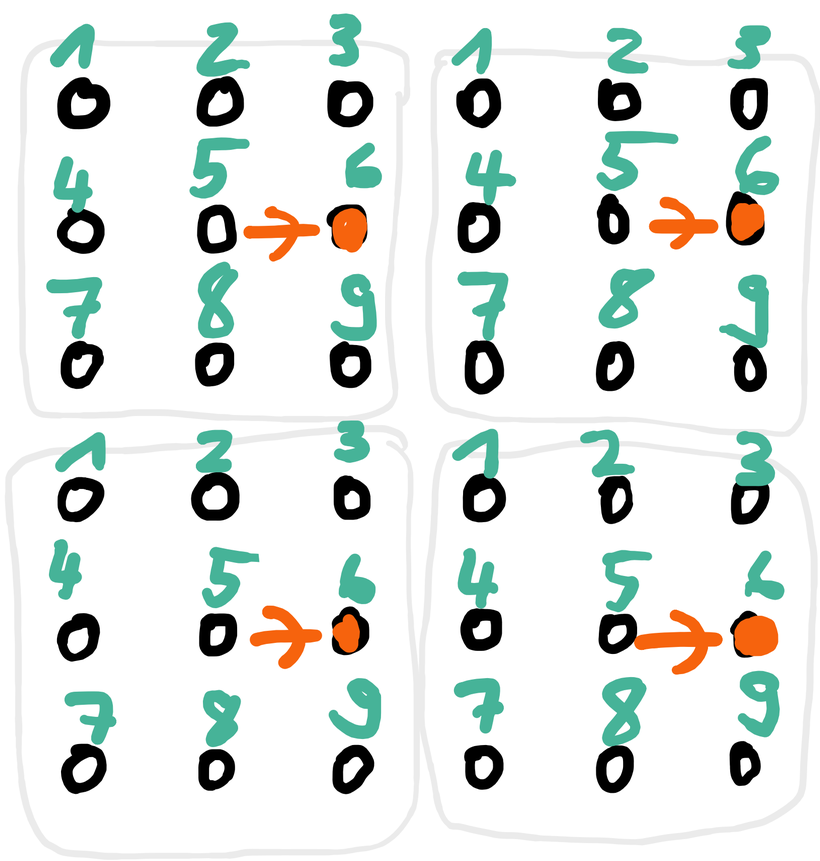
OPTIMIZING THE DIRAC OPERATOR
FULLY COALESCED NEIGHBORS?

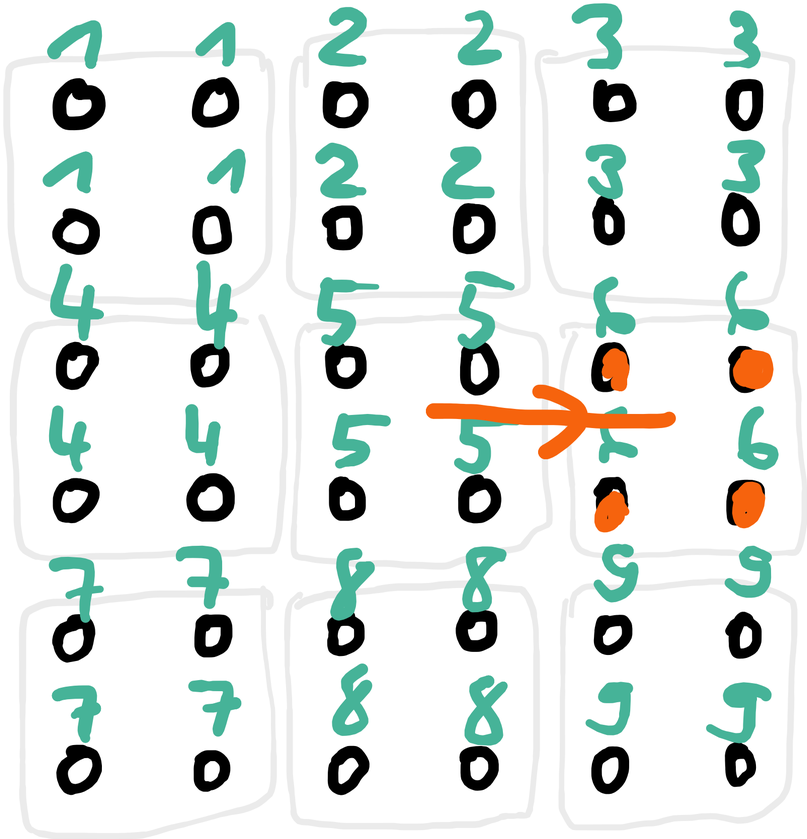




OUTPUT SITES

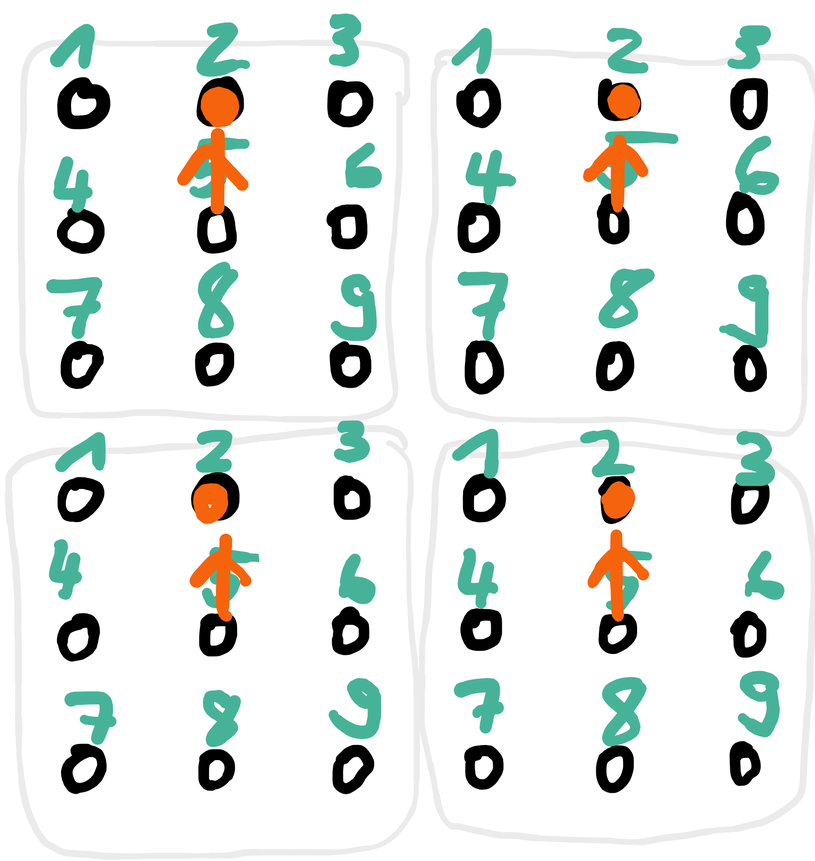
OPTIMIZING THE DIRAC OPERATOR
FULLY COALESCED NEIGHBORS?

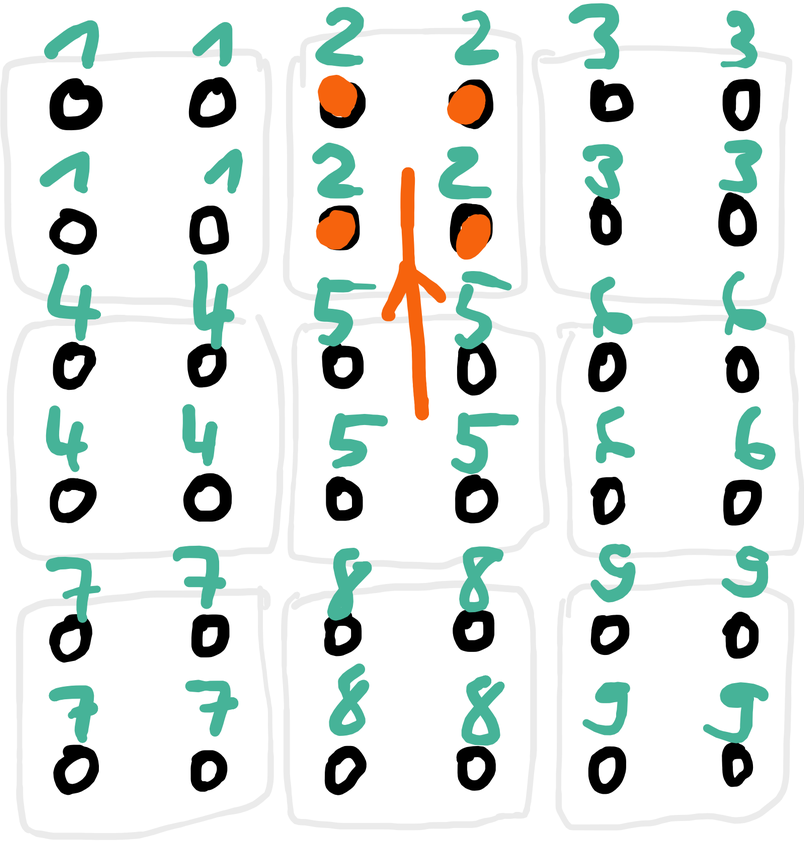




RIGHT NEIGHBORS

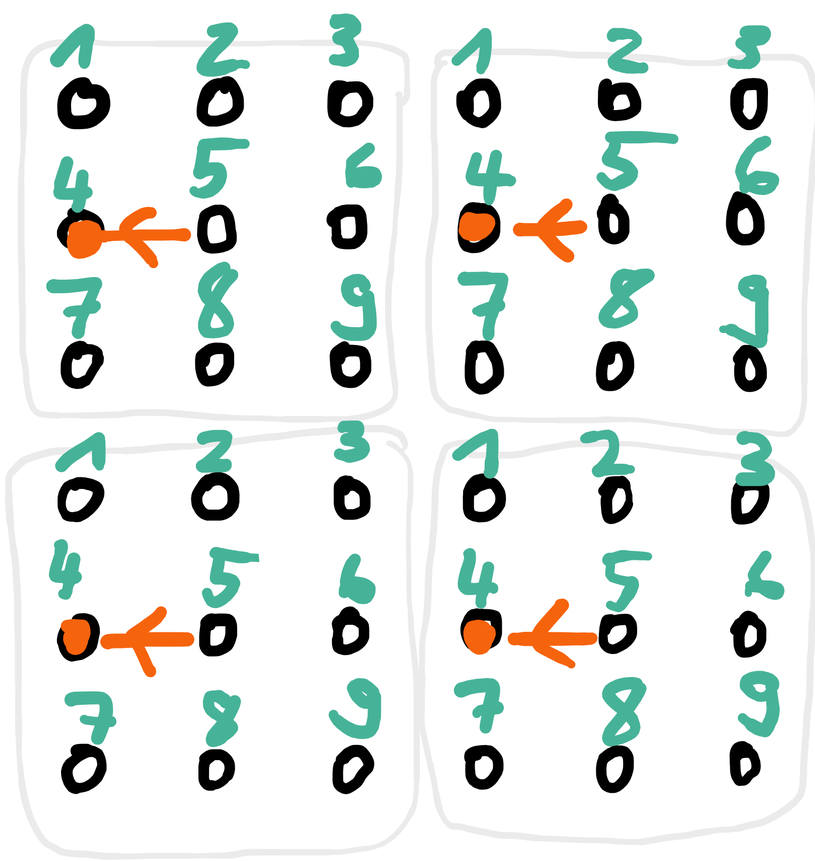
OPTIMIZING THE DIRAC OPERATOR
FULLY COALESCED NEIGHBORS?

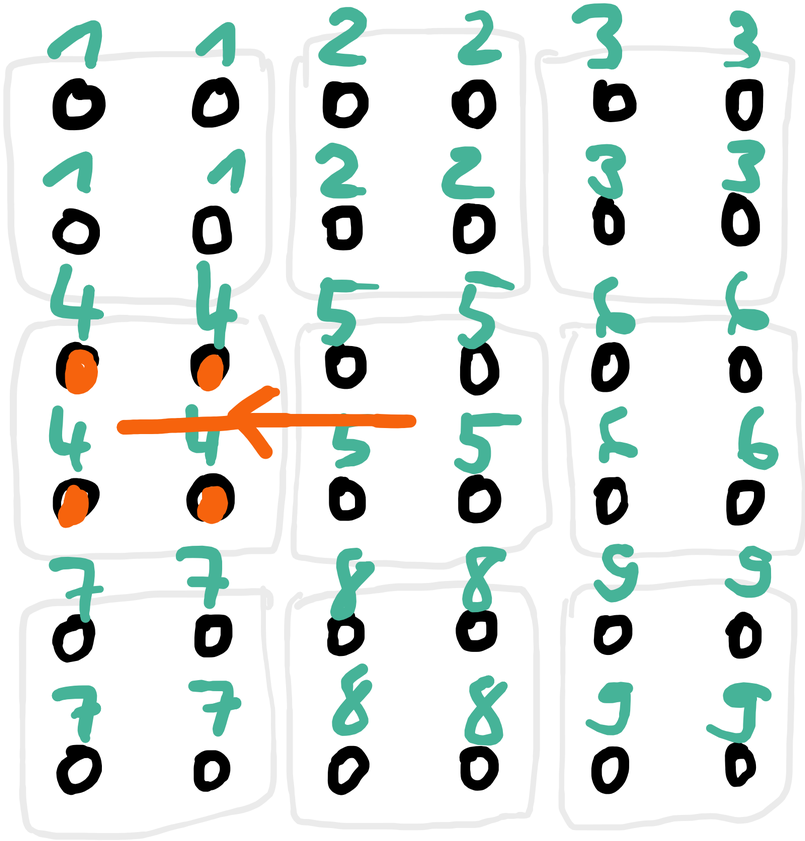




UP
NEIGHBORS

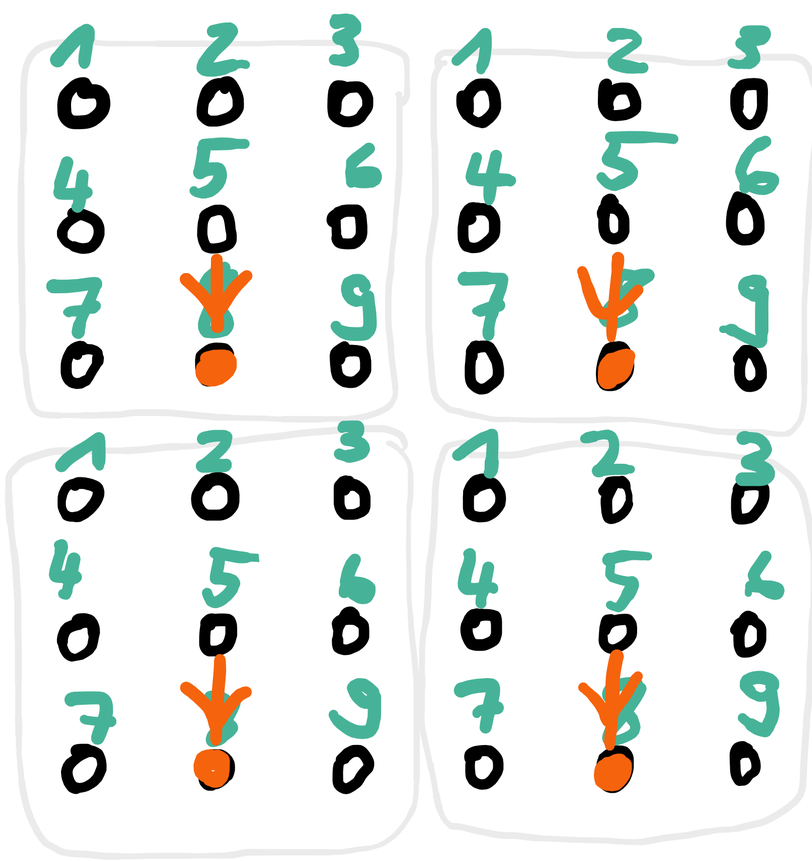
OPTIMIZING THE DIRAC OPERATOR
FULLY COALESCED NEIGHBORS?

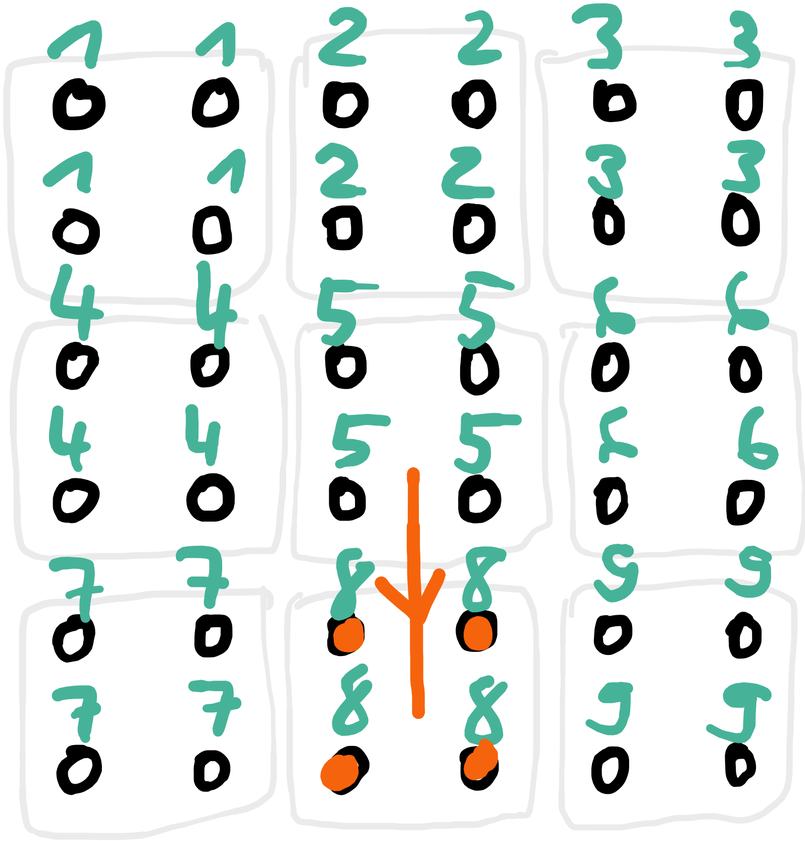




LEFT NEIGHBORS

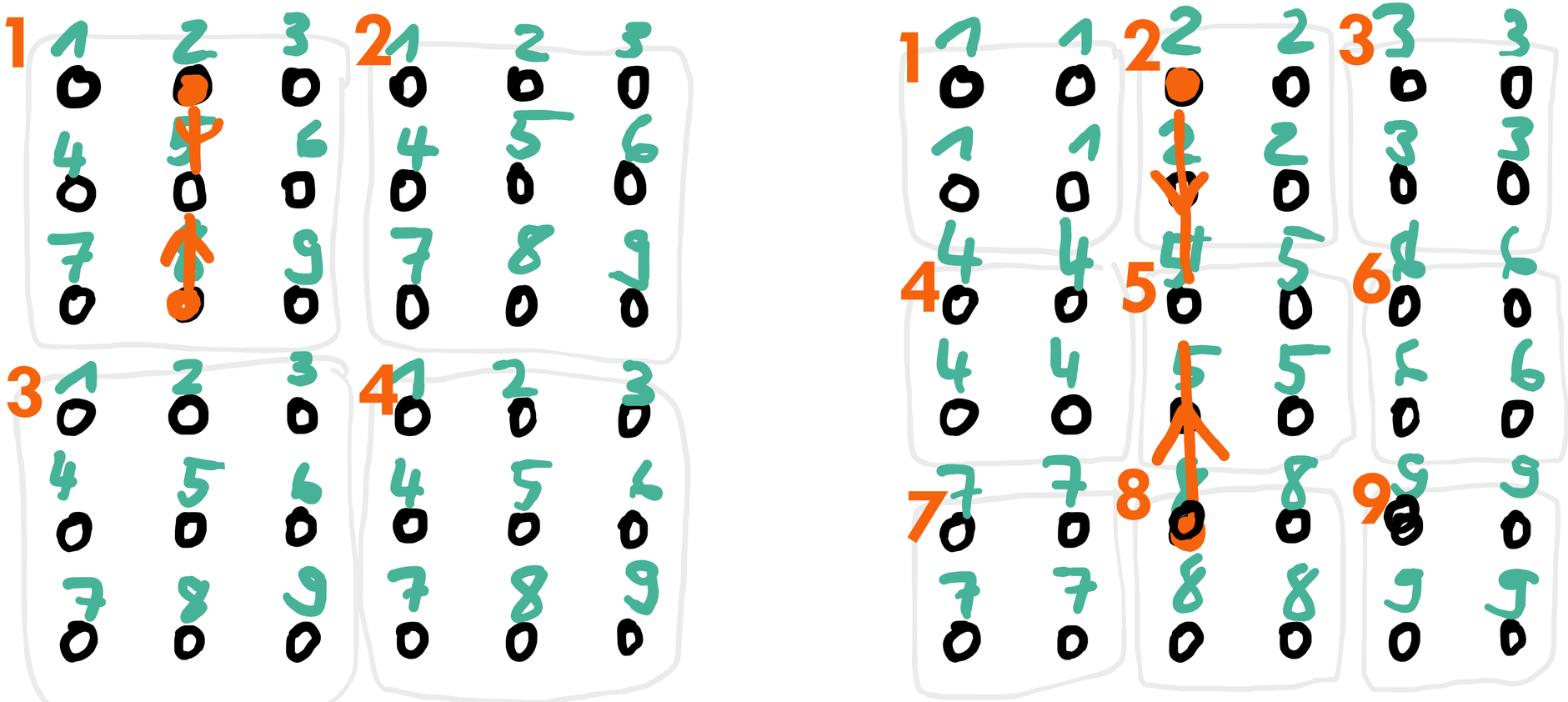
OPTIMIZING THE DIRAC OPERATOR
FULLY COALESCED NEIGHBORS?





DOWN NEIGHBORS

OPTIMIZING THE DIRAC OPERATOR PROBLEM: NO REUSE AGAIN!



Exact payoff: The time saved by exact coalescence is absorbed in lower L2 cache hit rate

REGISTER SPILLING

OPTIMIZING THE DIRAC OPERATOR REGISTER SPILLING



$$D_{y,x}\Psi_x$$

- Site Parallelization
-
-

Optimize the reading of the off-diagonal elements as explained above

Most of the time very efficient

OPTIMIZING THE DIRAC OPERATOR REGISTER SPILLING



$$D_{y,x}\Psi_x$$

- Site Parallelization
-
-

Optimize the reading of the off-diagonal elements as explained above

Most of the time very efficient

OPTIMIZING THE DIRAC OPERATOR REGISTER SPILLING



$$D_{ab,y,x}\Psi_{x,a}$$

1. Site parallelization
2. D_R and site parallelization

+ Less memory per thread
- Duplicated memory reads from global memory (spinor components)
- Great cache locality for duplicated reads, immediate reuse

OPTIMIZING THE DIRAC OPERATOR REGISTER SPILLING



$$D_{ab,y,x}\Psi_{x,a}$$

1. Site parallelization
2. D_R and site parallelization

+ Less memory per thread
- Duplicated memory reads from global memory (spinor components)
- Great cache locality for duplicated reads, immediate reuse

Sometimes faster than (1.), much faster for large D_R

OPTIMIZING THE DIRAC OPERATOR REGISTER SPILLING

$$D^{ts}_{ab,y,x}\Psi^{s}_{x,a}$$

1. Site parallelization
2. D_R and site parallelization
3. Spin, D_R and site parallelization

+ Even less memory per thread
- Even more duplicated reads
- Branch divergence

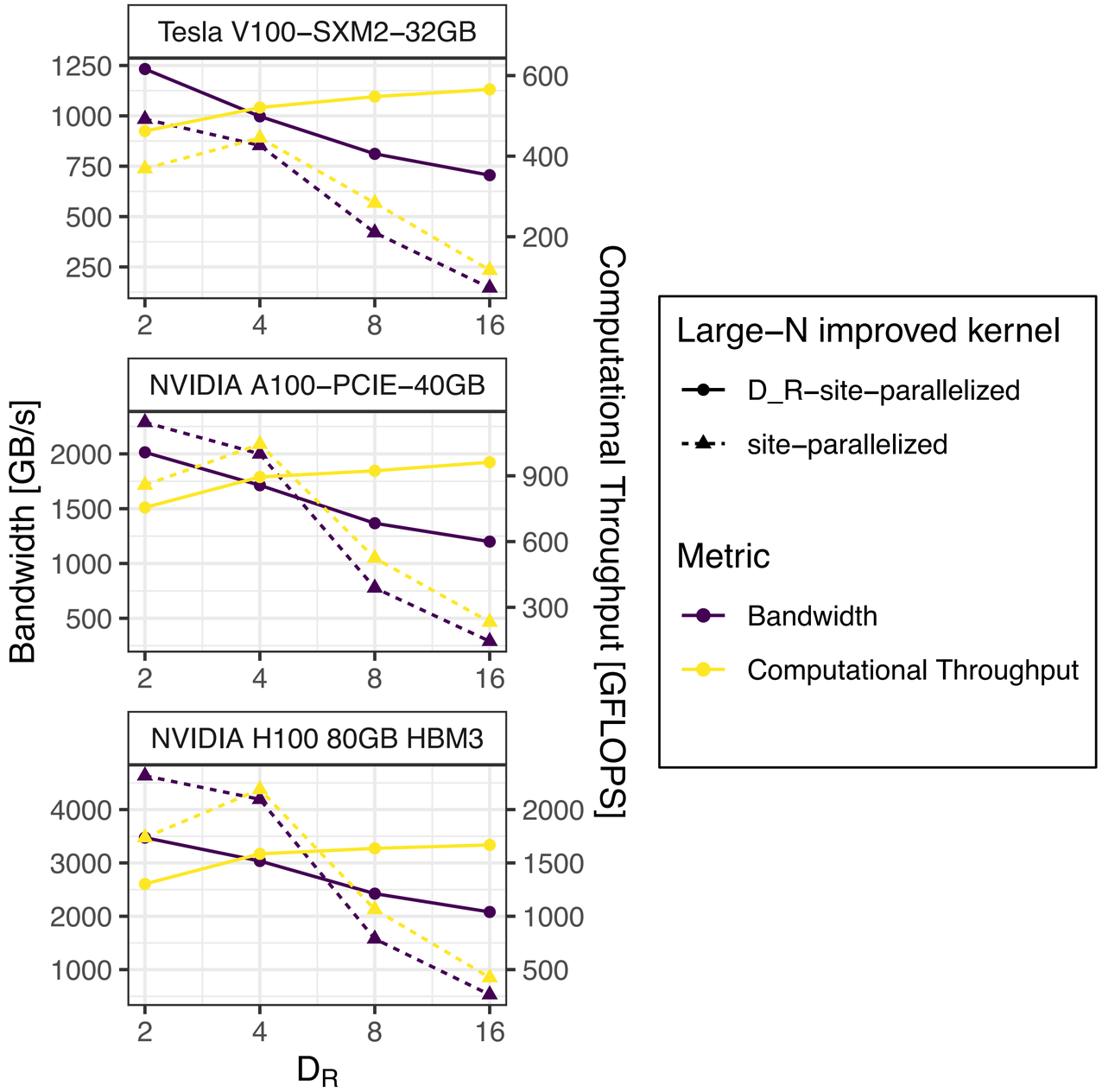
[SM et al., 2025, 2503.06721]

OPTIMIZING THE DIRAC OPERATOR REGISTER SPILLING

The D_R-site-parallelized kernel saves local memory

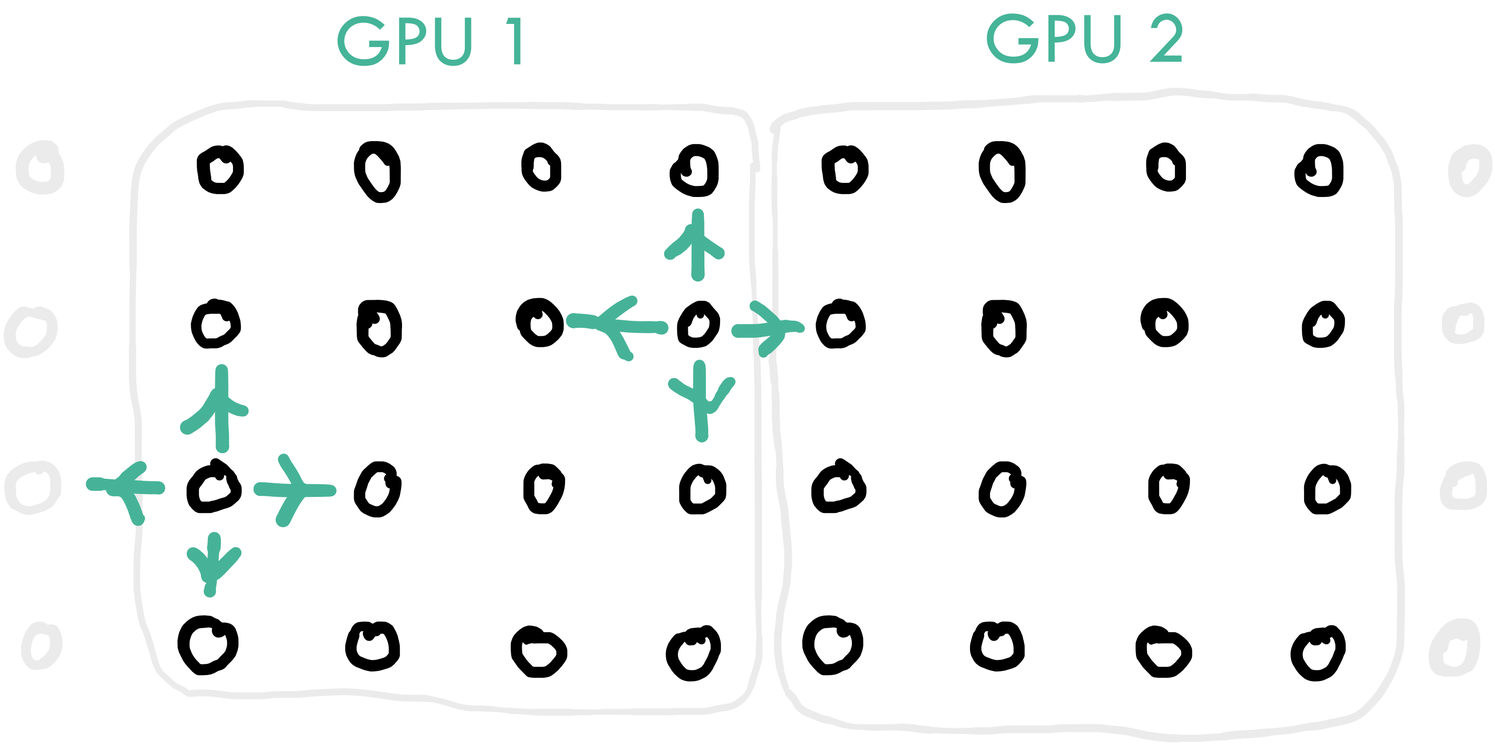
It scales better for large-N or D_R

On the newest GPU one needs very large D_R to outperform the standard kernel

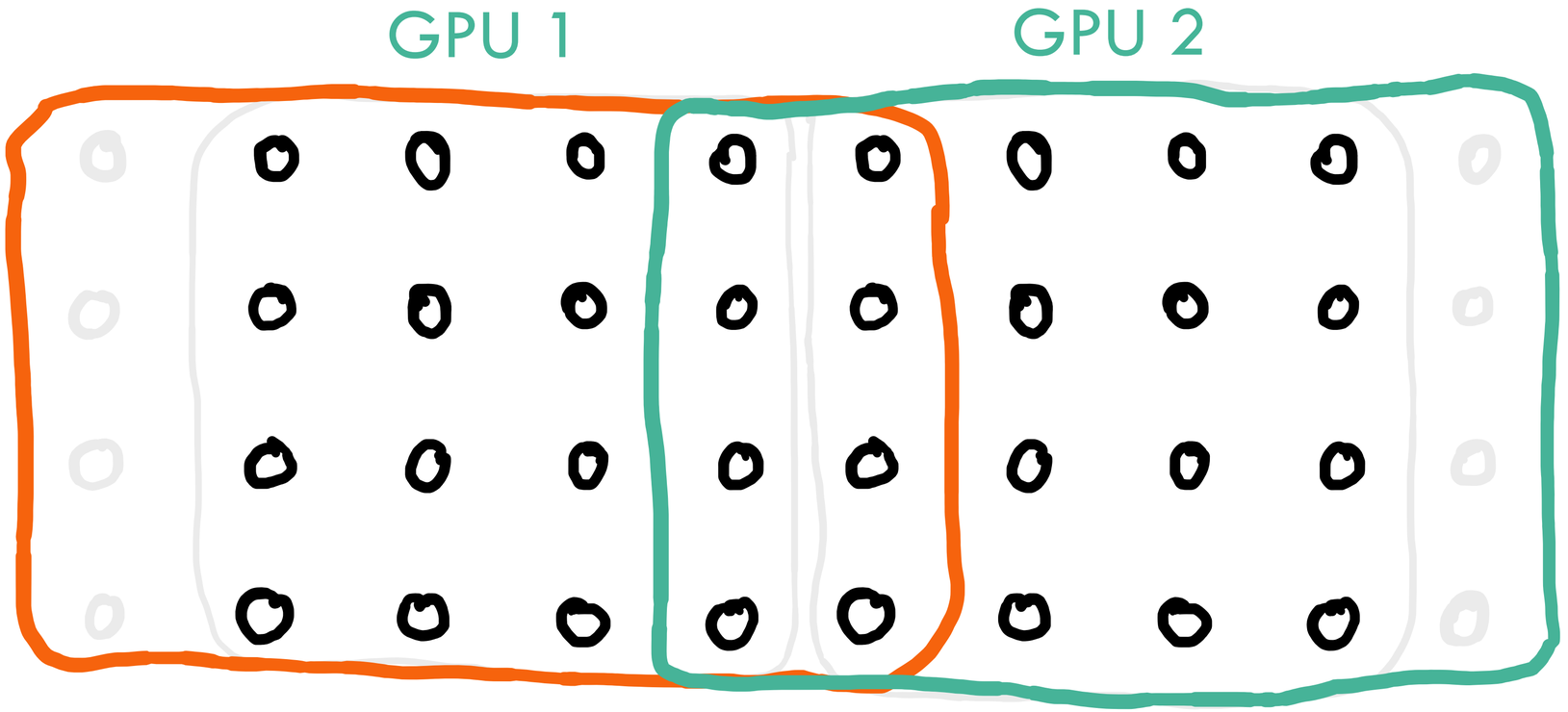


PARALLELIZING FOR MULTIPLE GPUS

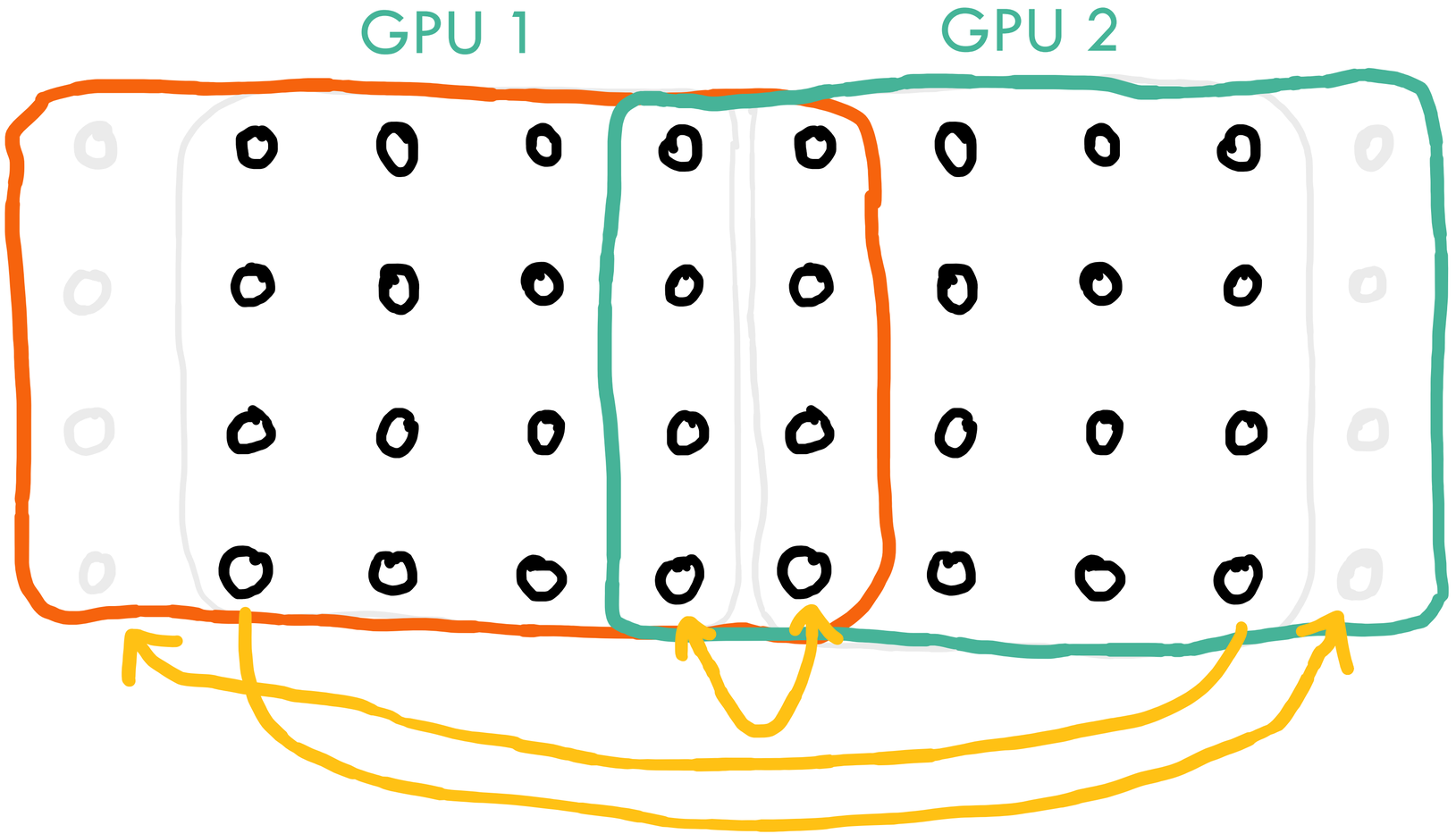
USING MULTIPLE GPUS AT ONCE
HIDING COMMUNICATION LATENCIES



USING MULTIPLE GPUS AT ONCE
HIDING COMMUNICATION LATENCIES

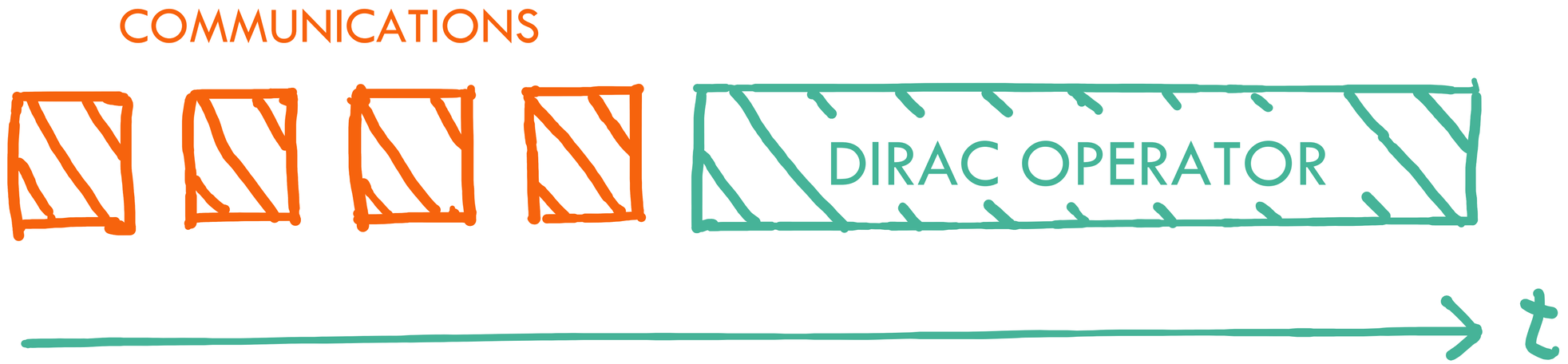


USING MULTIPLE GPUS AT ONCE
HIDING COMMUNICATION LATENCIES

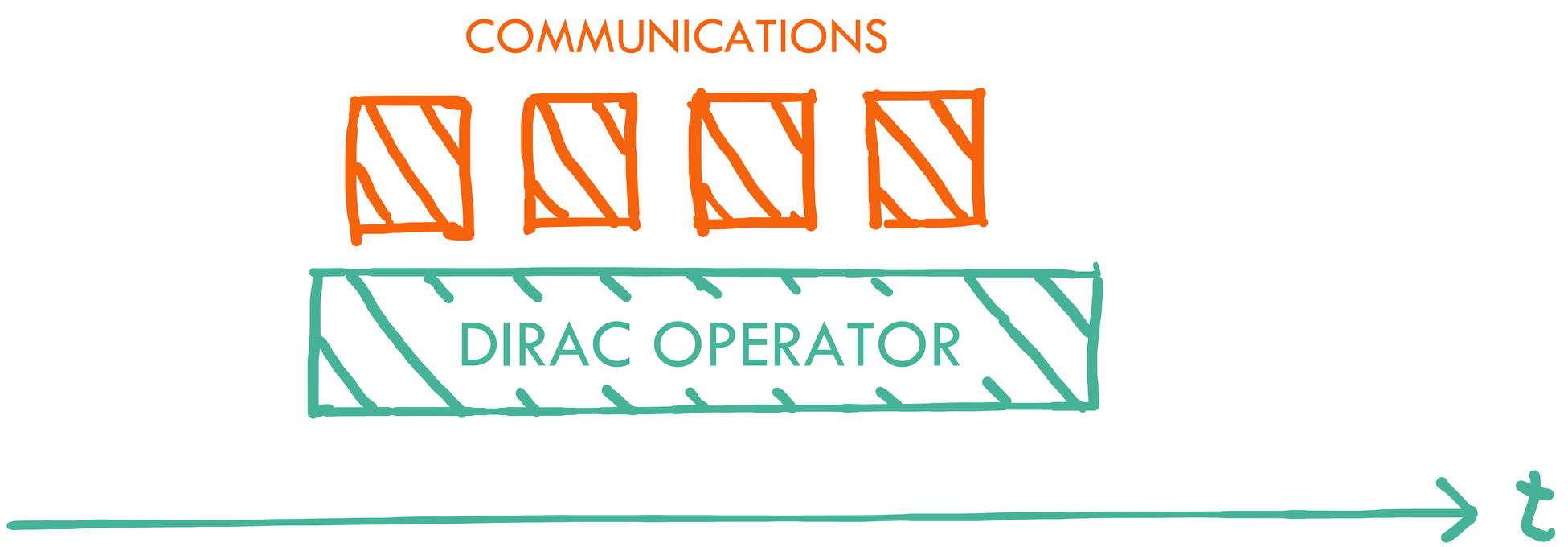


HIDING COMMUNICATION LATENCIES

USING MULTIPLE GPUS AT ONCE
HIDING COMMUNICATION LATENCIES



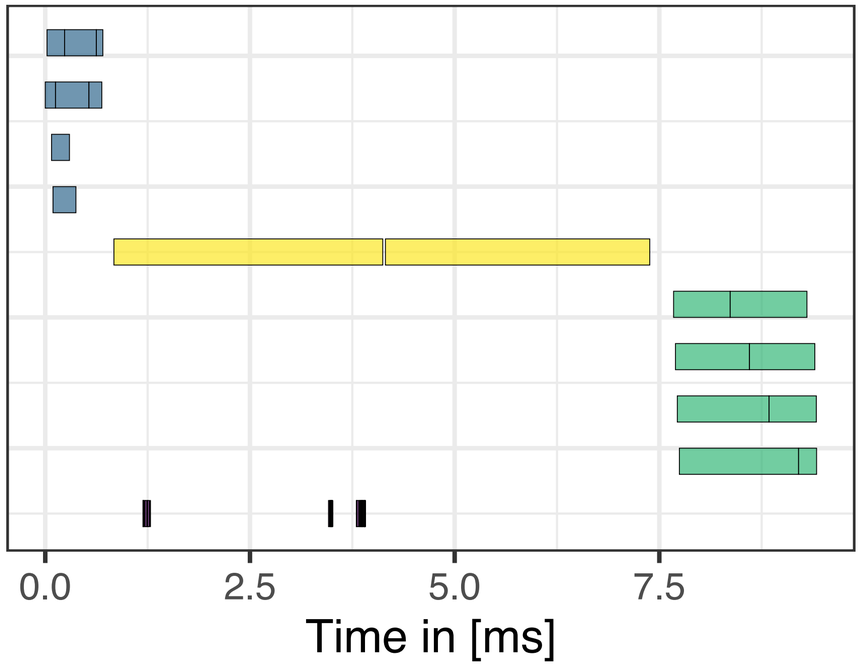
USING MULTIPLE GPUS AT ONCE
HIDING COMMUNICATION LATENCIES



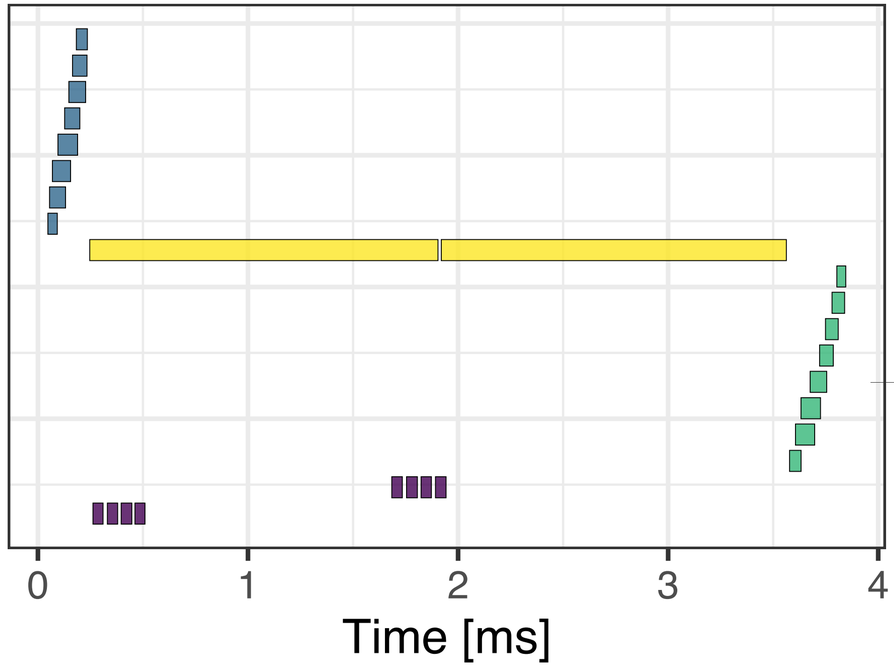
BENCHMARKING

TRACING PROFILER

MI250X (AMD/HIP)



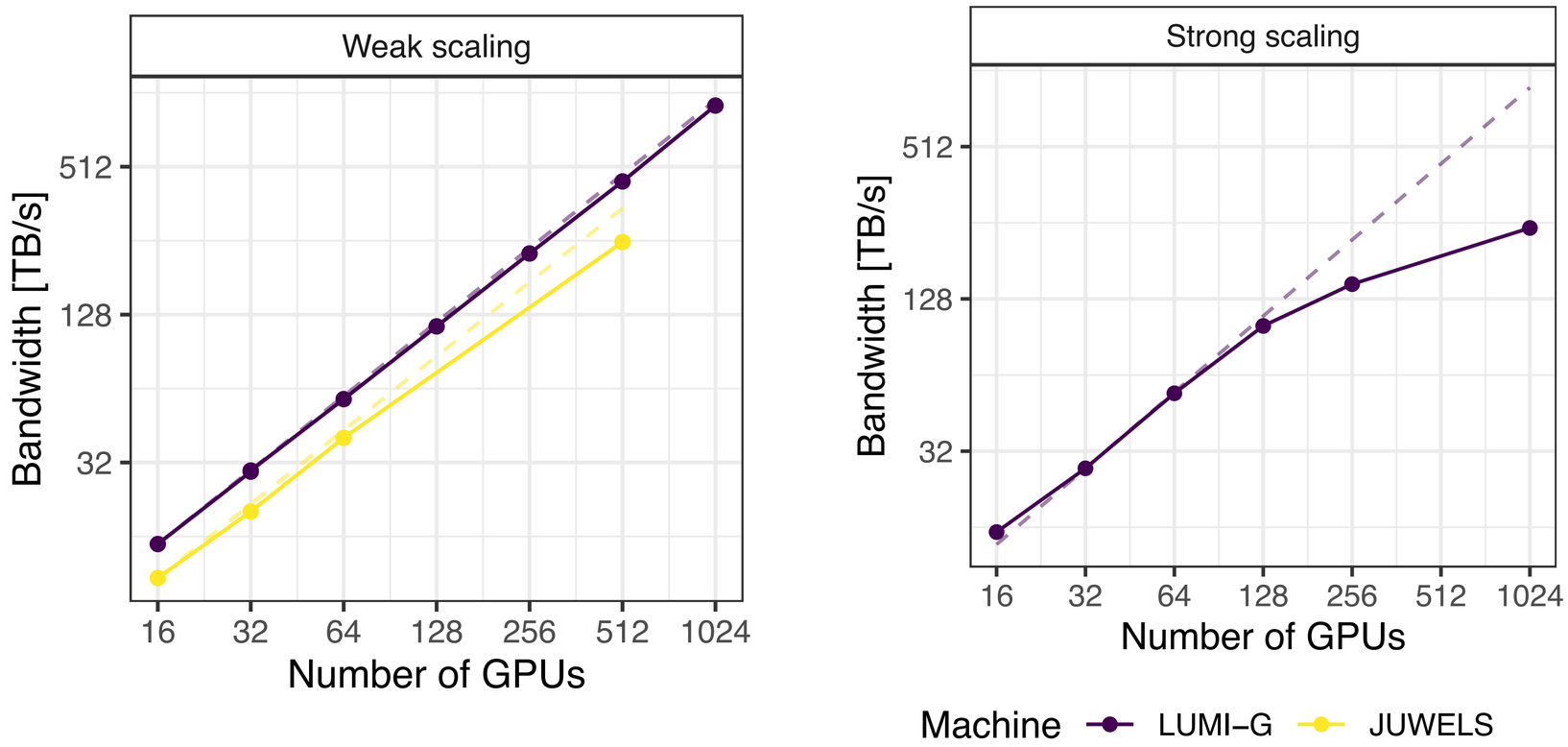
NVIDIA H100 (CUDA)



Using a 48^4 local lattice, parallelized in two dimensions on four GPUs/GCDs respectively

[SM et al, 2024, 2411.18511, PoS]

SCALING TO HUGE WORKLOADS



Thank you for your attention!