Large Scale Parallel Lattice Boltzmann Model of Dendritic Growth

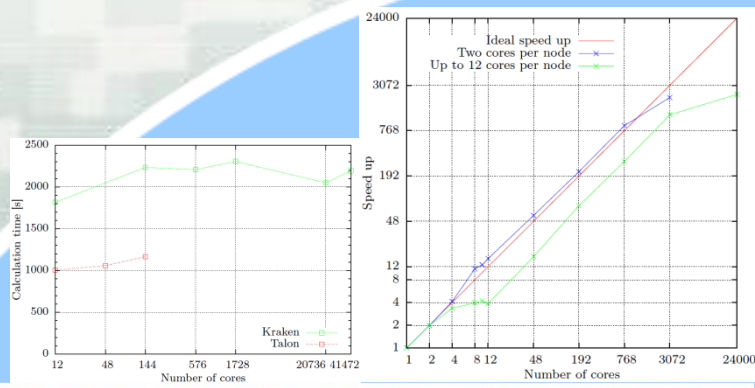
Bohumir Jelinek

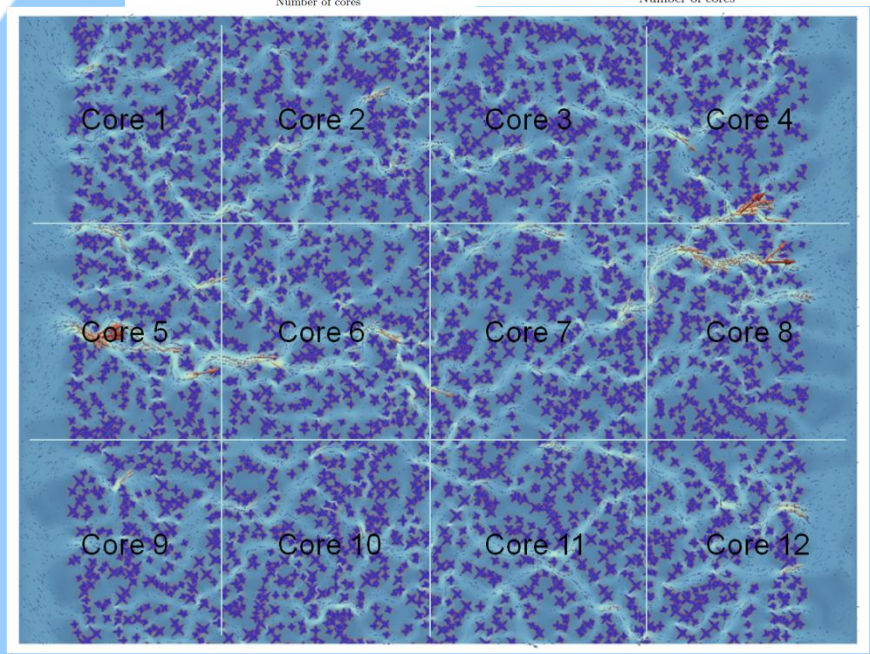
Mohsen Eshraghi

Sergio Felicelli

CAVS, Mississippi State University

March 3-7, 2013 – San Antonio, Texas



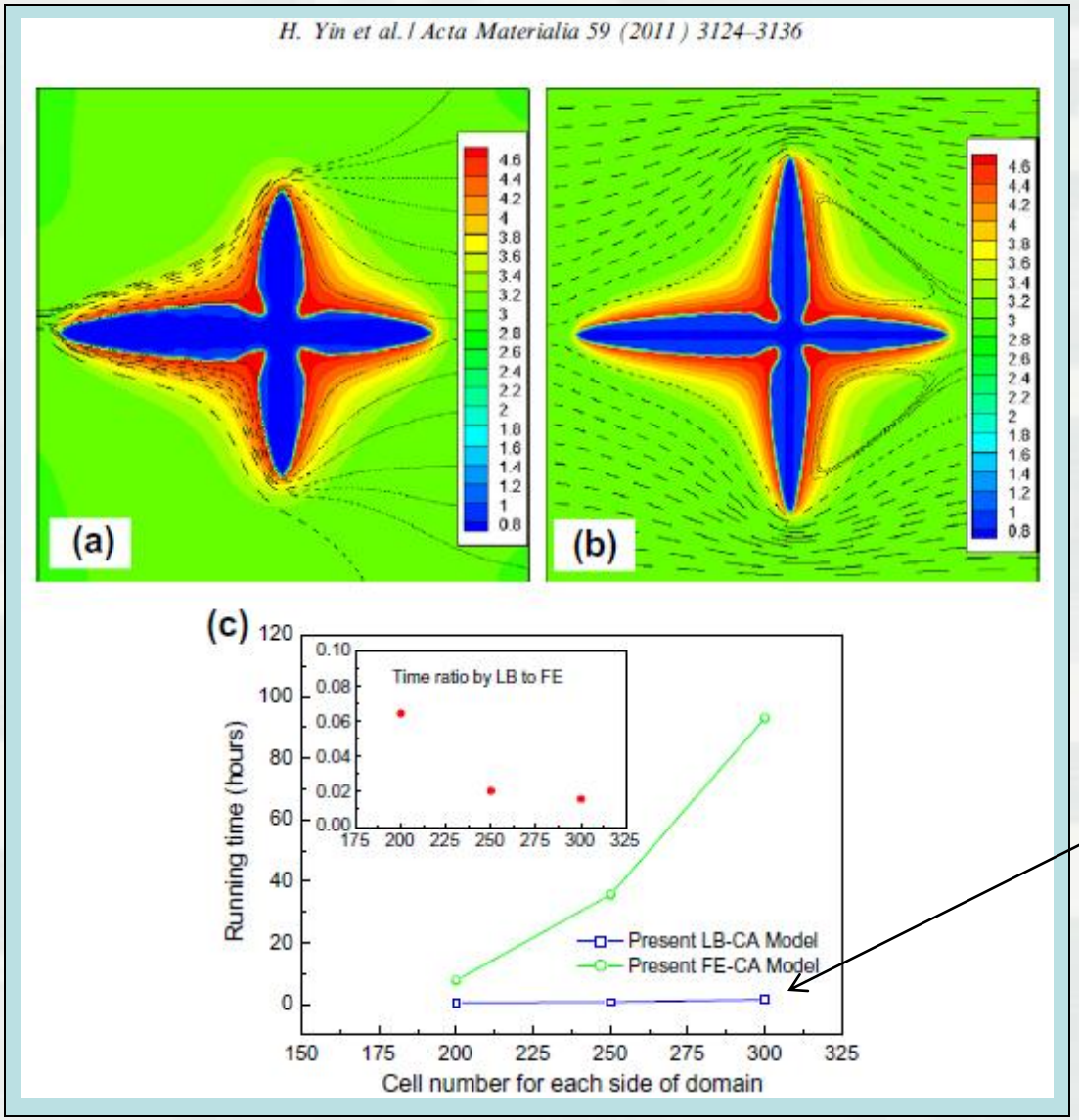


MISSISSIPPI STATE UNIVERSITY CAVS

US Army Corps of Engineers

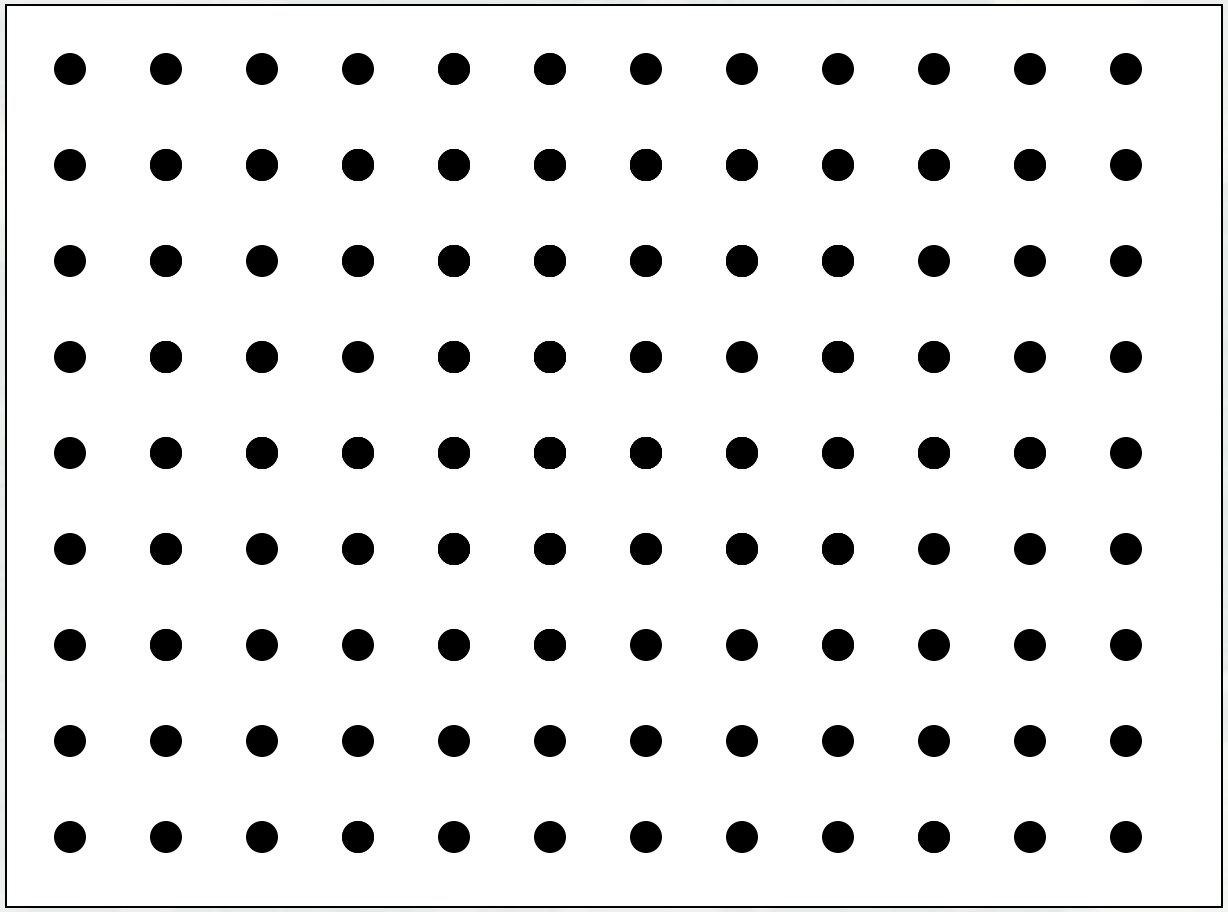
BUILDING STRONG®

Why LBM-CA?



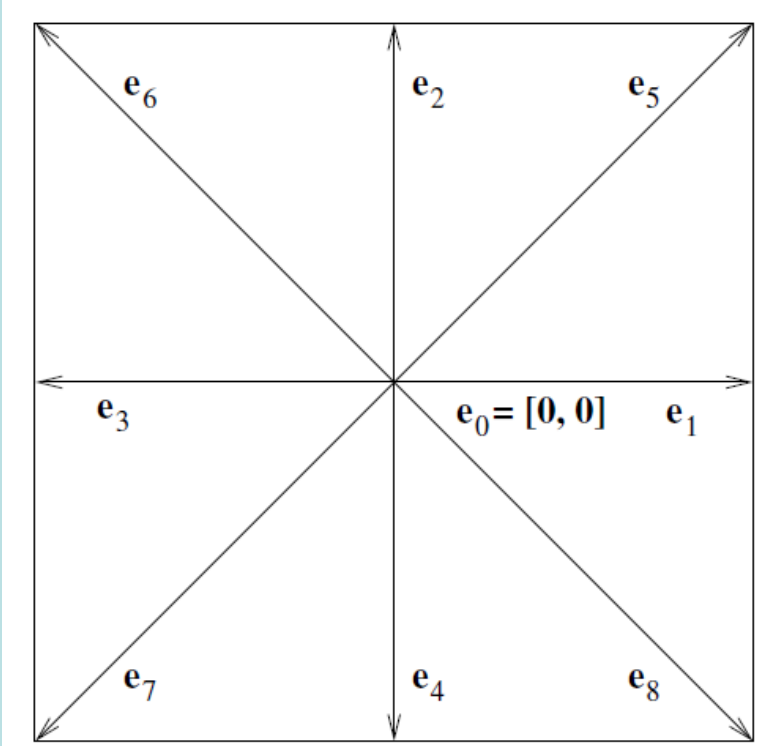
When the fluid flow around solidifying dendrites is considered, lattice Boltzmann method is faster than alternatives

Lattice Boltzmann method



Lattice-Boltzmann method (LBM) calculates time evolution of a quantity of interest governed by a partial differential equation subject to given initial and boundary conditions at regularly spaced nodes .

Lattice Boltzmann Method



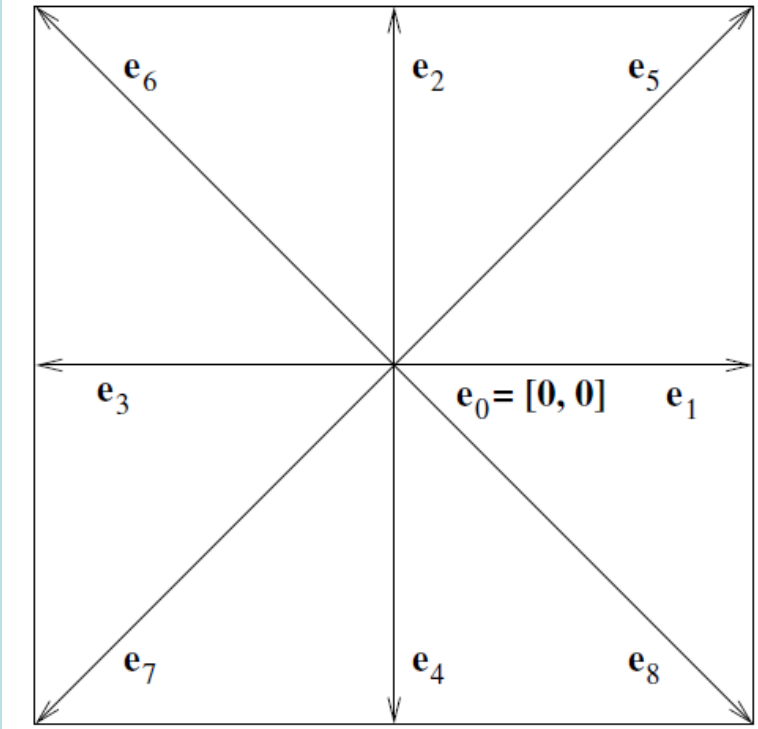
D2Q9 lattice

Each node has 9 distribution functions f_i representing portion

of the mass density moving in the lattice direction $\boldsymbol{e_i}$

$$\rho = \sum_{i=0}^{8} f_i, \qquad \rho \boldsymbol{u} = \sum_{i=0}^{8} f_i \boldsymbol{e}_i$$

Lattice Boltzmann Method



D2Q9 lattice

Each node has 9 distribution functions f_i representing portion

of the mass density moving in the lattice direction $\boldsymbol{e_i}$

$$\rho = \sum_{i=0}^{8} f_i, \quad \rho \boldsymbol{u} = \sum_{i=0}^{8} f_i \boldsymbol{e}_i$$

$$f_i(\boldsymbol{r} + \boldsymbol{e}_i \Delta t, t + \Delta t) = f_i(\boldsymbol{r}, t) + \frac{1}{\tau_u}\left(f_i^{eq}(\boldsymbol{r}, t) - f_i(\boldsymbol{r}, t)\right)$$

Evolution of the distribution functions

$$f_i(\boldsymbol{r} + \boldsymbol{e}_i \Delta t, t + \Delta t) = f_i(\boldsymbol{r}, t) + \frac{1}{\tau_u}\left(f_i^{eq}(\boldsymbol{r}, t) - f_i(\boldsymbol{r}, t)\right)$$

Evolution of the distribution functions

For each lattice direction $\boldsymbol{e}_i$, i=0..8

$$f_i(\boldsymbol{r} + \boldsymbol{e}_i\Delta t, t + \Delta t) = f_i(\boldsymbol{r}, t) + \frac{1}{\tau_u}\left(f_i^{eq}(\boldsymbol{r}, t) - f_i(\boldsymbol{r}, t)\right)$$

Evolution of the distribution functions

For each lattice direction $\boldsymbol{e}_i$, i=0..8

$$f_i(\boldsymbol{r} + \boldsymbol{e}_i\Delta t, t + \Delta t) = f_i(\boldsymbol{r}, t) + \frac{1}{\tau_u}\left(f_i^{eq}(\boldsymbol{r}, t) - f_i(\boldsymbol{r}, t)\right)$$

Collision:

Adjusts the distribution function to approach equilibrium distribution

Evolution of the distribution functions

For each lattice direction $\boldsymbol{e}_i$, i=0..8

$$f_i(\boldsymbol{r} + \boldsymbol{e}_i\Delta t, t + \Delta t) = f_i(\boldsymbol{r}, t) + \frac{1}{\tau_u}\left(f_i^{eq}(\boldsymbol{r}, t) - f_i(\boldsymbol{r}, t)\right)$$

Streaming:

Shifts each distribution function to the neighboring node

Collision:

Adjusts the distribution function to approach equilibrium distribution

Evolution of the distribution functions

$$\nu = \frac{\tau_u - 0.5}{3}\frac{\Delta x^2}{\Delta t}$$

For each lattice direction $\boldsymbol{e}_i$, i=0..8

τ_u ~kin. viscosity

$$f_i(\boldsymbol{r} + \boldsymbol{e}_i\Delta t, t + \Delta t) = f_i(\boldsymbol{r}, t) + \frac{1}{\tau_u}\left(f_i^{eq}(\boldsymbol{r}, t) - f_i(\boldsymbol{r}, t)\right)$$

Streaming:

Shifts each distribution function to the neighboring node

Collision:

Adjusts the distribution function to approach equilibrium distribution

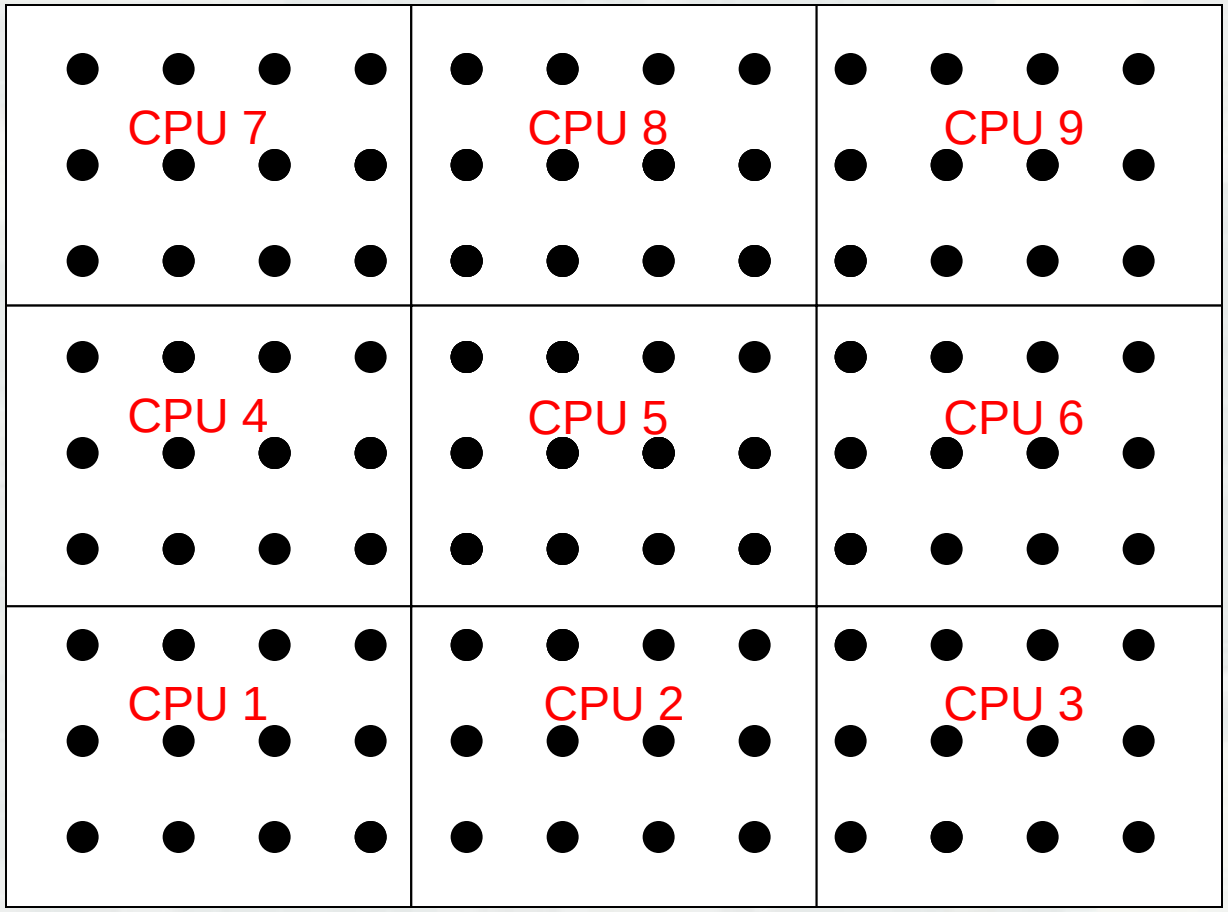
Equilibrium distribution function

$$f_i^{\mathrm{eq}}(\boldsymbol{r}) = w_i \rho(\boldsymbol{r}) \left(1 + 3\frac{\boldsymbol{e}_i \cdot \boldsymbol{u}(\boldsymbol{r})}{c^2} + \frac{9}{2}\frac{(\boldsymbol{e}_i \cdot \boldsymbol{u}(\boldsymbol{r}))^2}{c^4} - \frac{3}{2}\frac{\boldsymbol{u}(\boldsymbol{r}) \cdot \boldsymbol{u}(\boldsymbol{r})}{c^2} \right)$$

$$w_i = \begin{cases} 4/9 & i = 0 \\ 1/9 & i = 1, 2, 3, 4 \\ 1/36 & i = 5, 6, 7, 8 \end{cases}$$

lattice velocity $c = \Delta x / \Delta t$

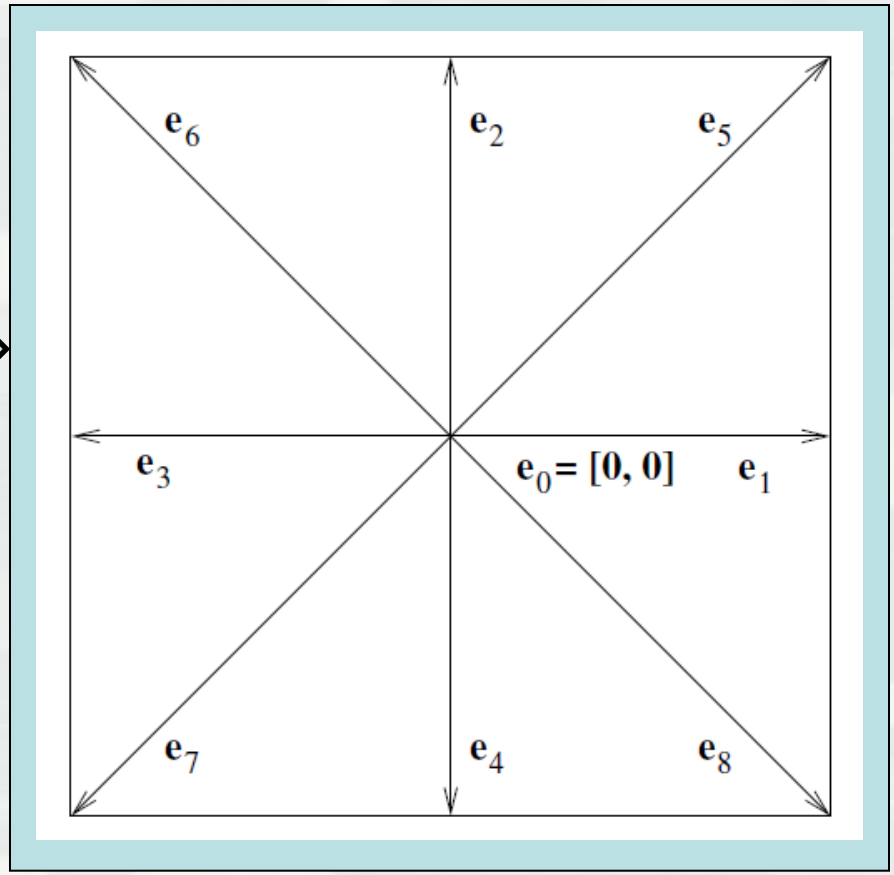
LBM parallelization



Spatial domain decomposition

LBM parallelization streaming

For each lattice direction $\boldsymbol{e}_i$
i=0..8



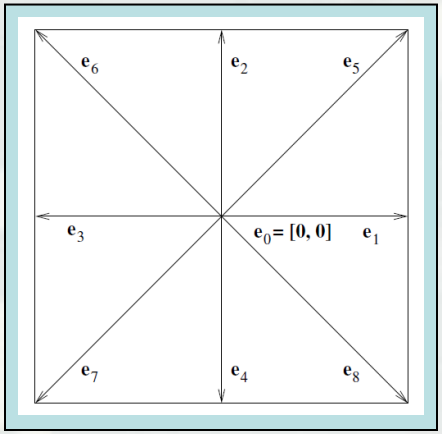
Streaming:

Shifts each distribution function to the neighboring node

$$f_i(\boldsymbol{r} + \boldsymbol{e}_i\Delta t, t + \Delta t) = f_i(\boldsymbol{r}, t) + \frac{1}{\tau_u}\left(f_i^{eq}(\boldsymbol{r}, t) - f_i(\boldsymbol{r}, t)\right)$$

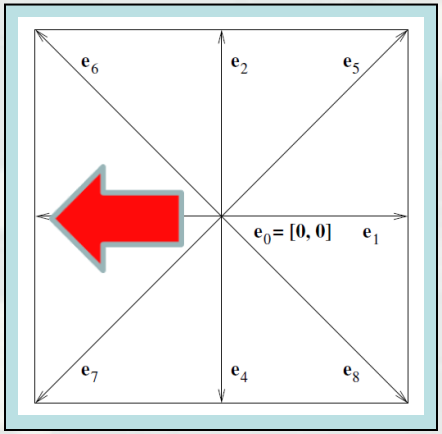
LBM parallelization – streaming

Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)



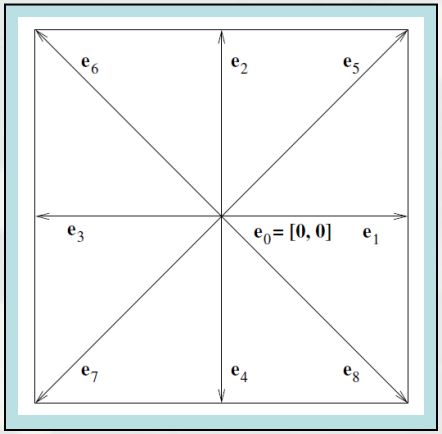
LBM parallelization – streaming

Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)



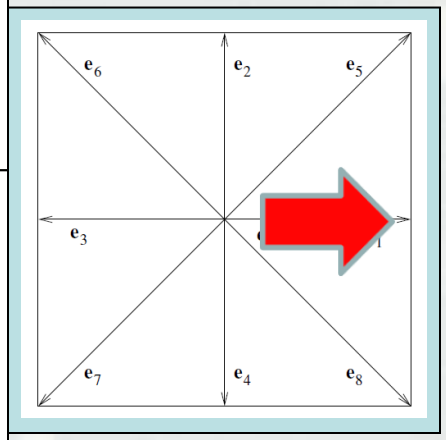
LBM parallelization – streaming

Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)



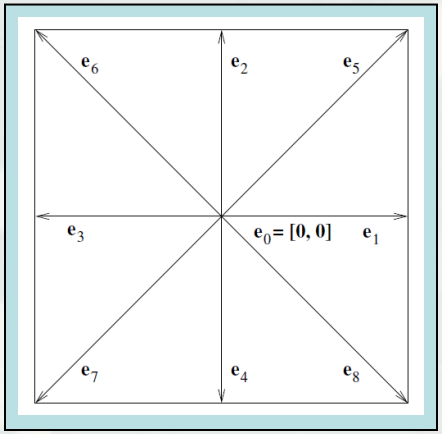
LBM parallelization – streaming

Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)



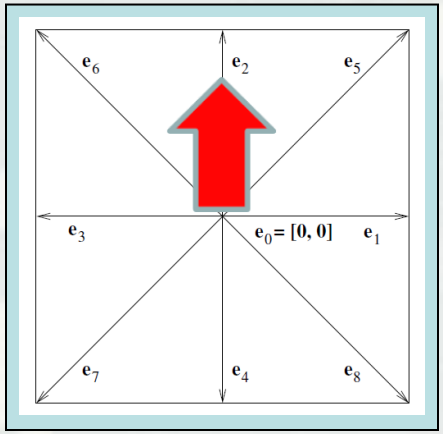
LBM parallelization – streaming

Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)



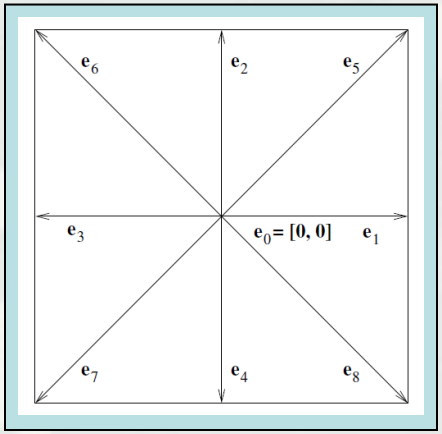
LBM parallelization – streaming

Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)



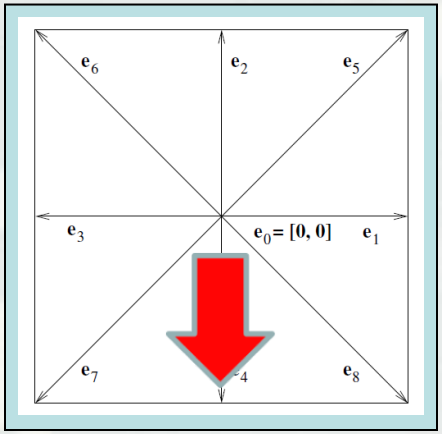
LBM parallelization – streaming

Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)



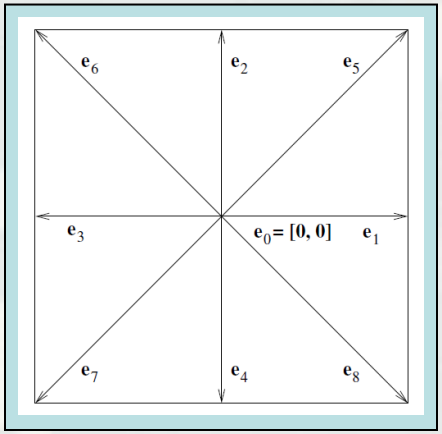
LBM parallelization – streaming

Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)

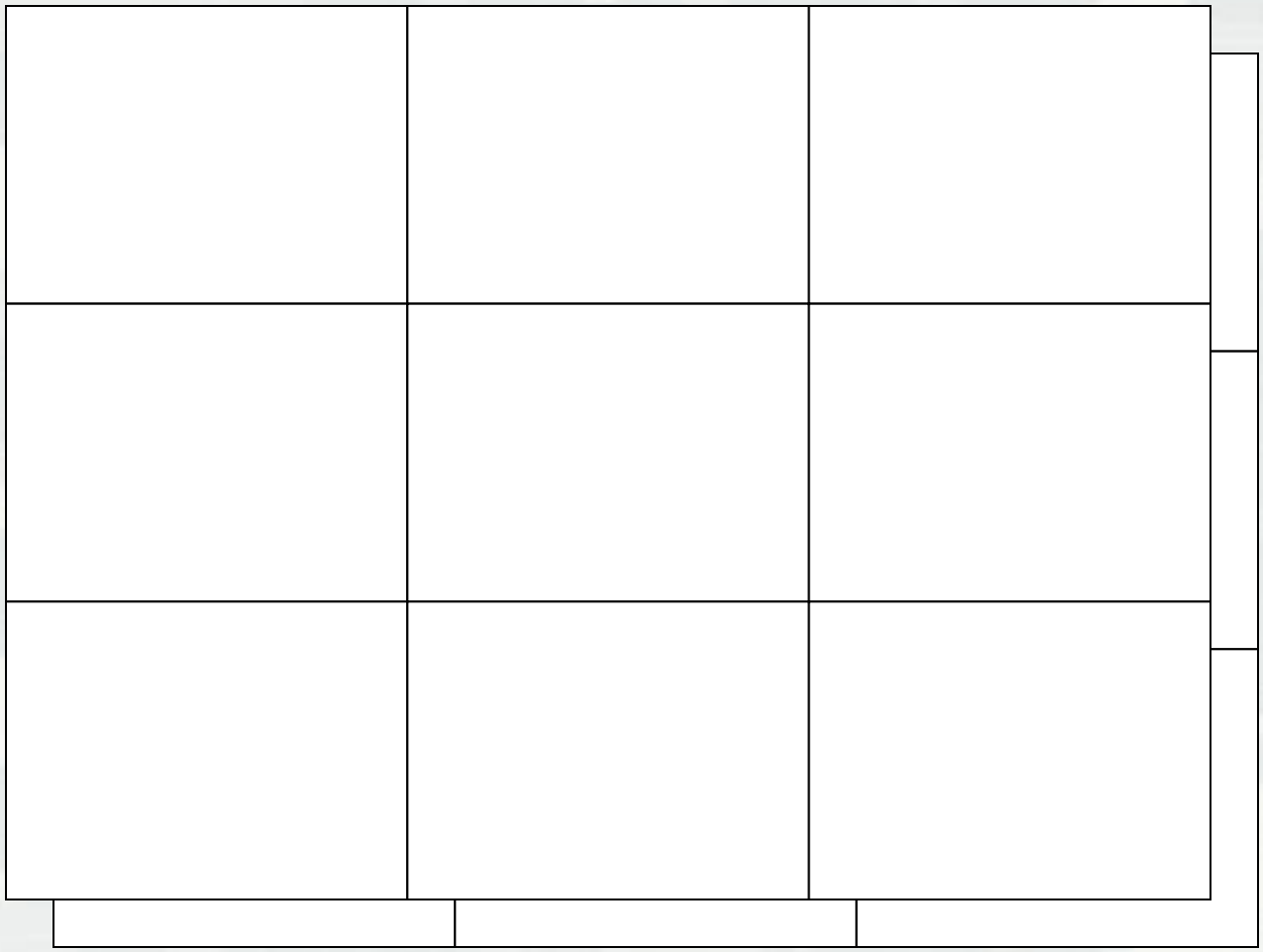


LBM parallelization – streaming

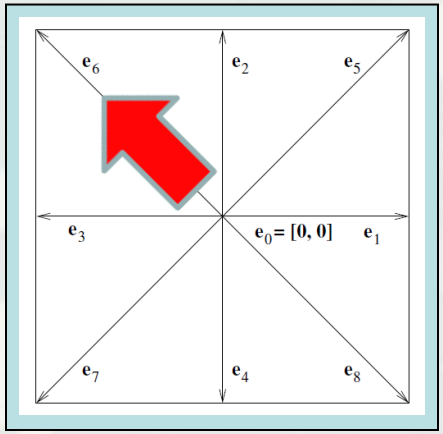
Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)



LBM parallelization – streaming

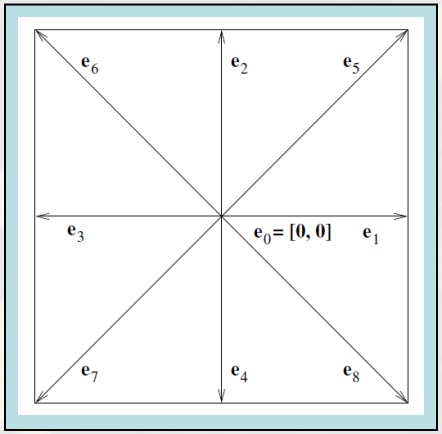


Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)

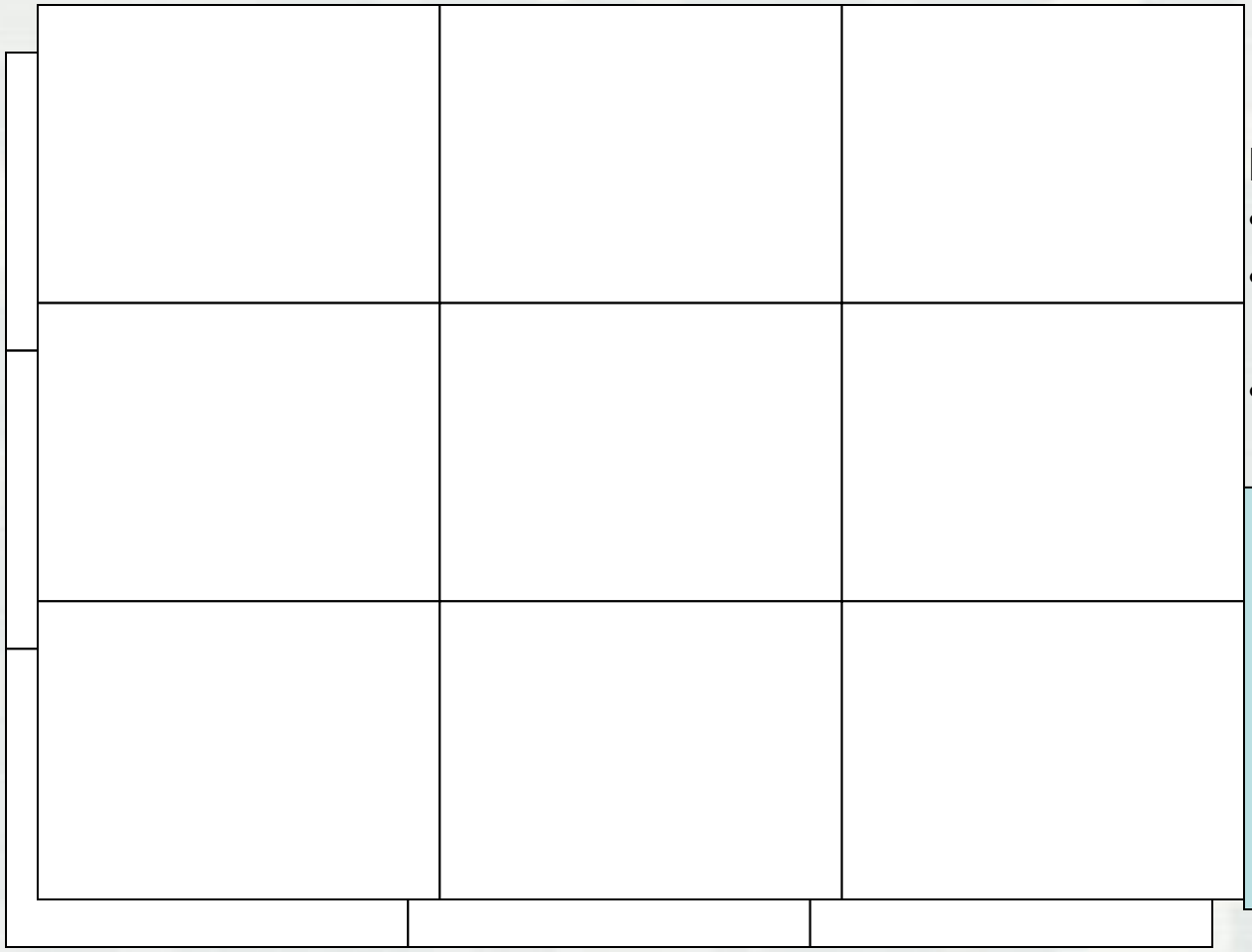


LBM parallelization – streaming

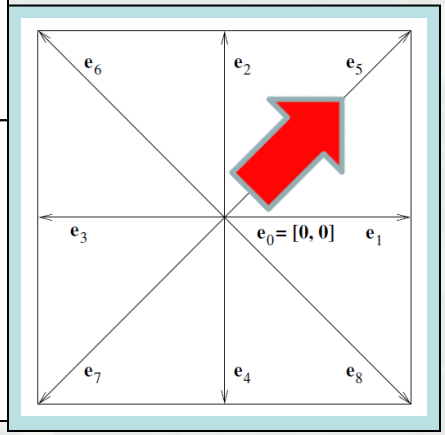
Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)



LBM parallelization – streaming

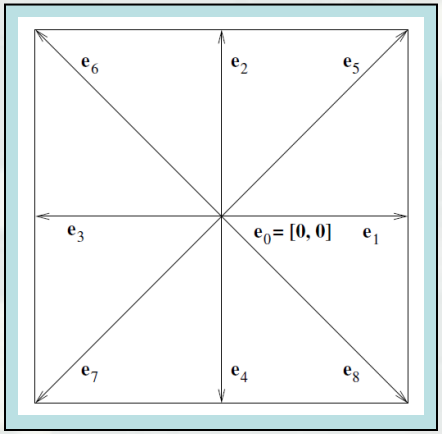


Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)

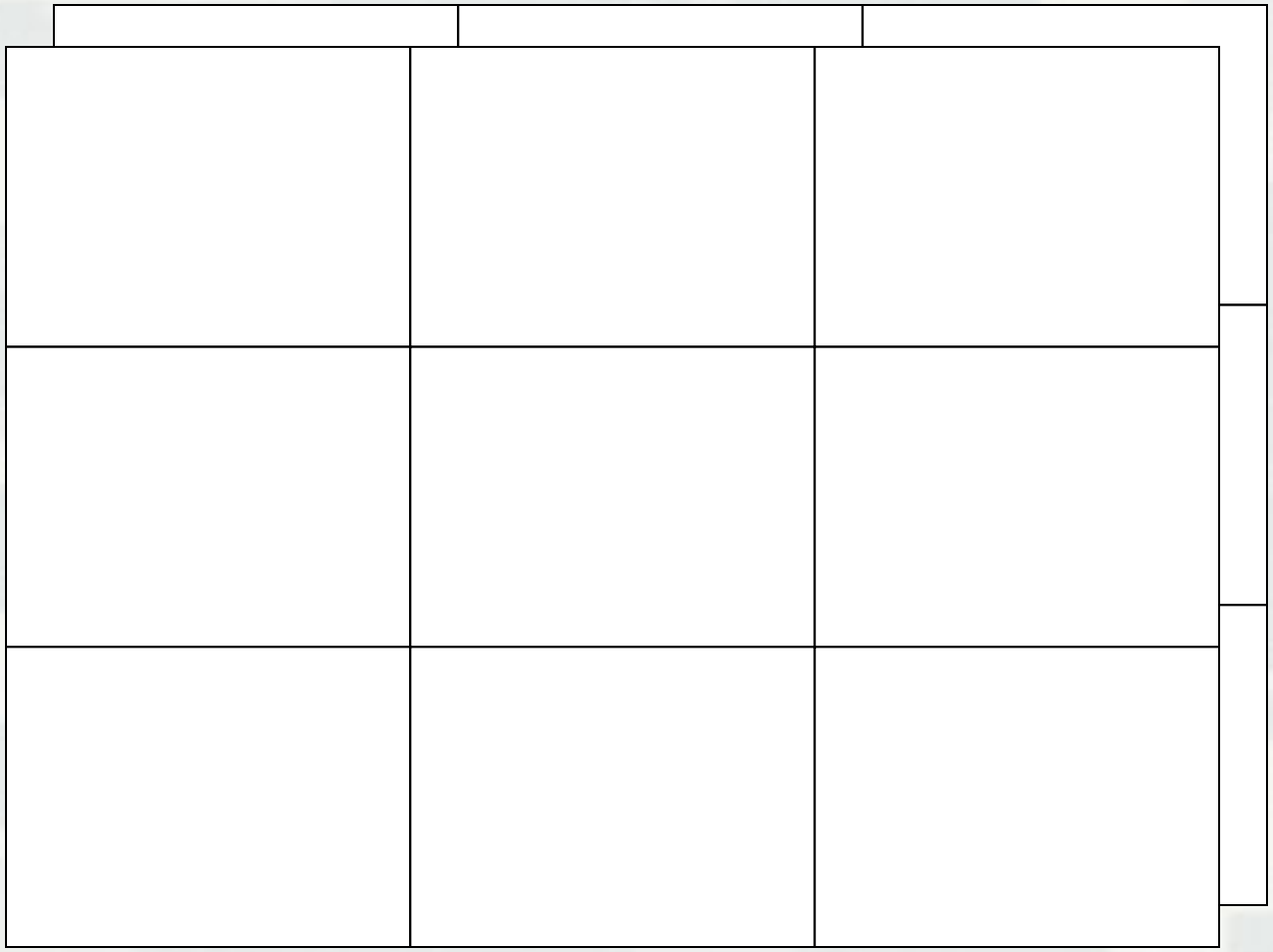


LBM parallelization – streaming

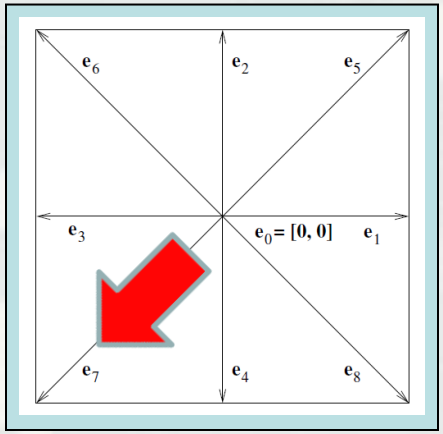
Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)



LBM parallelization – streaming

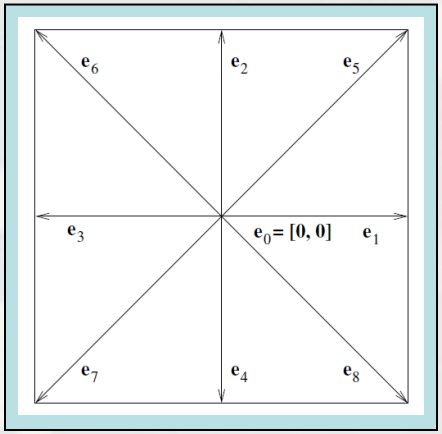


Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)

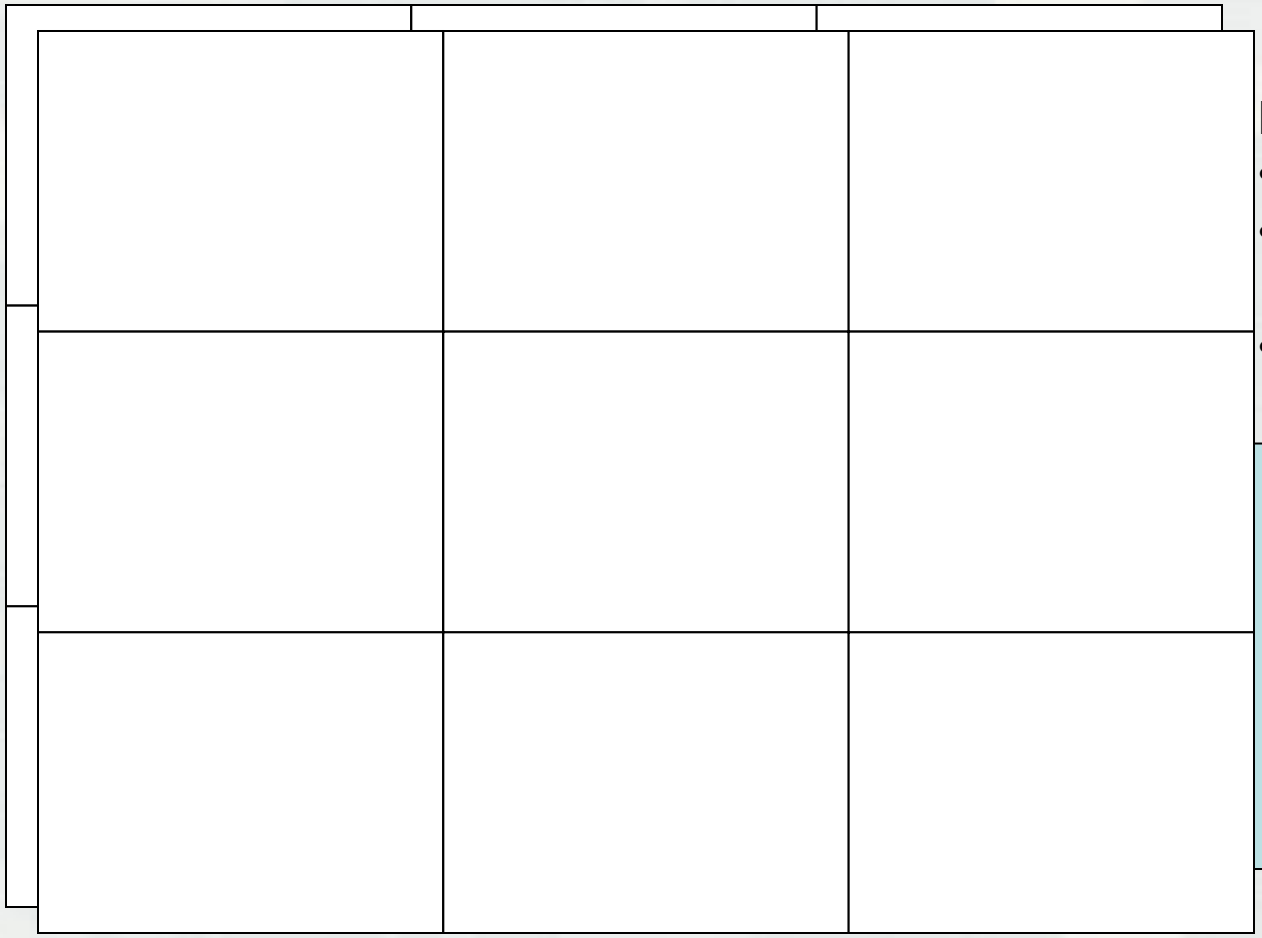


LBM parallelization – streaming

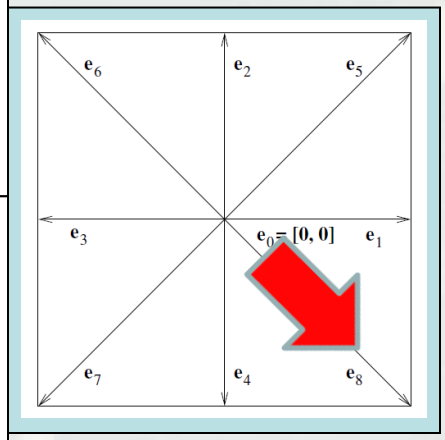
Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)



LBM parallelization – streaming



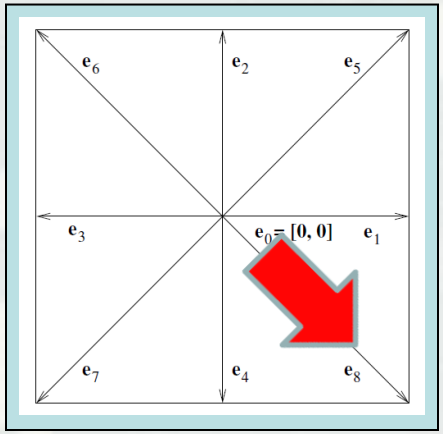
Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)



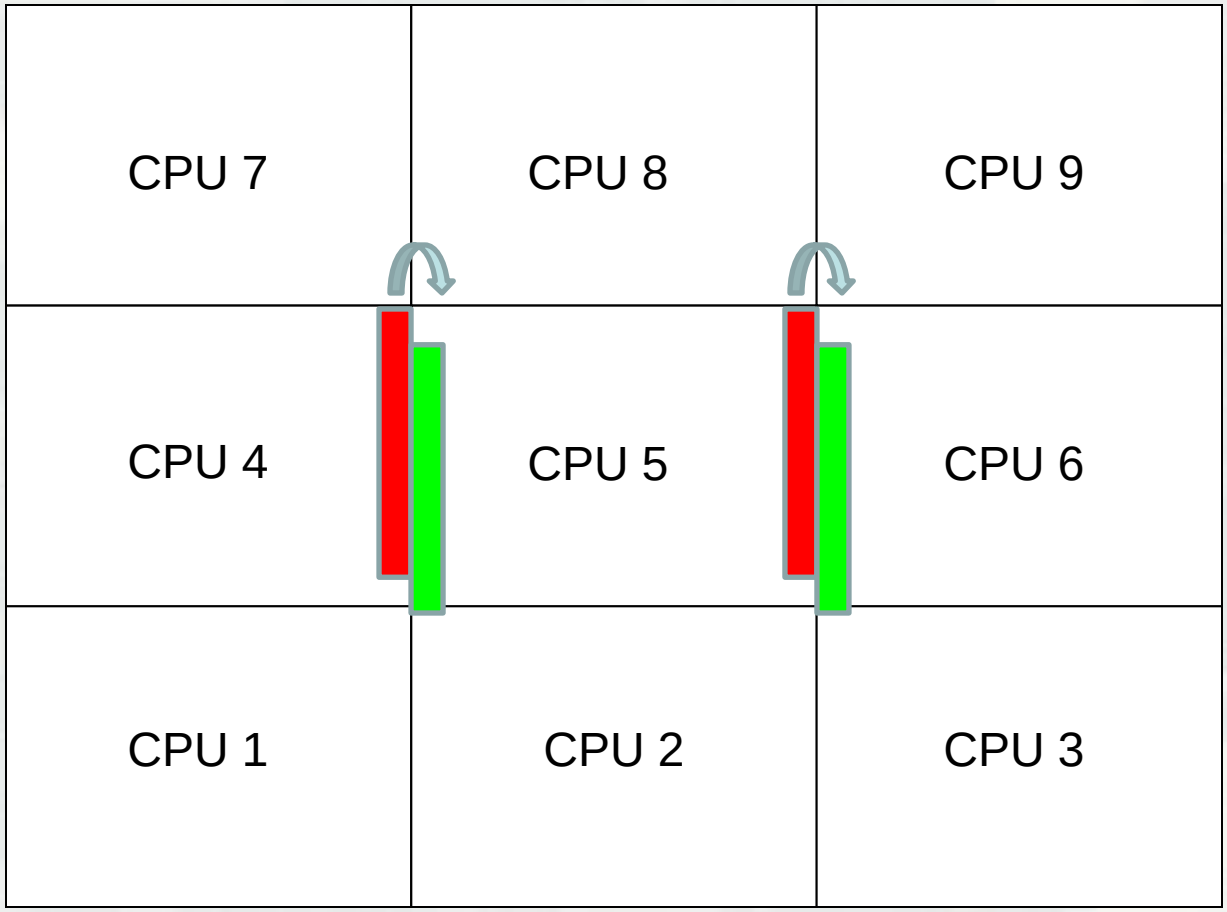
LBM parallelization – streaming

CPU 7	CPU 8	CPU 9
CPU 4	CPU 5	CPU 6
CPU 1	CPU 2	CPU 3

Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)



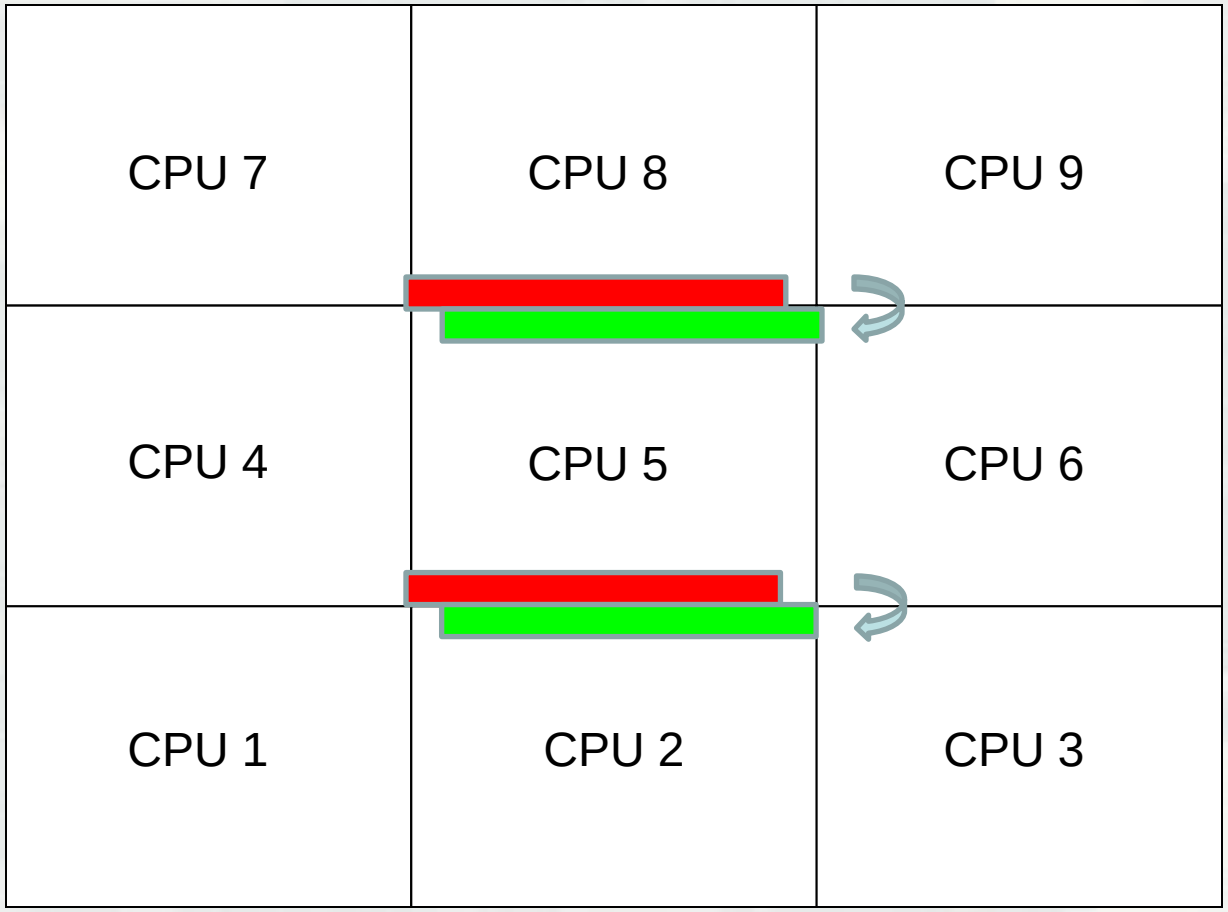
LBM parallelization – streaming



Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)

buffer=send
MPI_Sendrcv_replace(
  buffer, dst=6, src=4)
recv=buffer

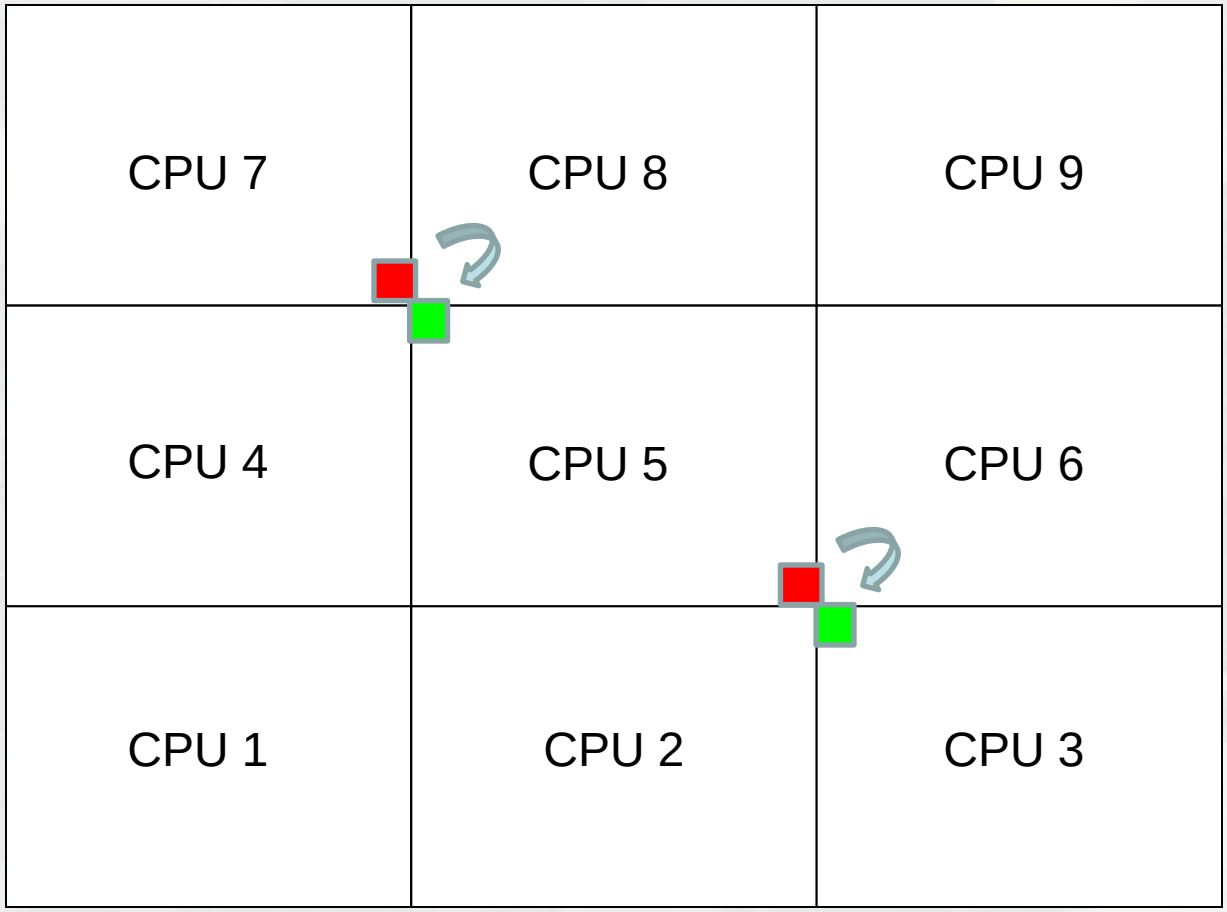
LBM parallelization – streaming



Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)

buffer=send
MPI_Sendrcv_replace(
  buffer, dst=2, src=8)
recv=buffer

LBM parallelization – streaming

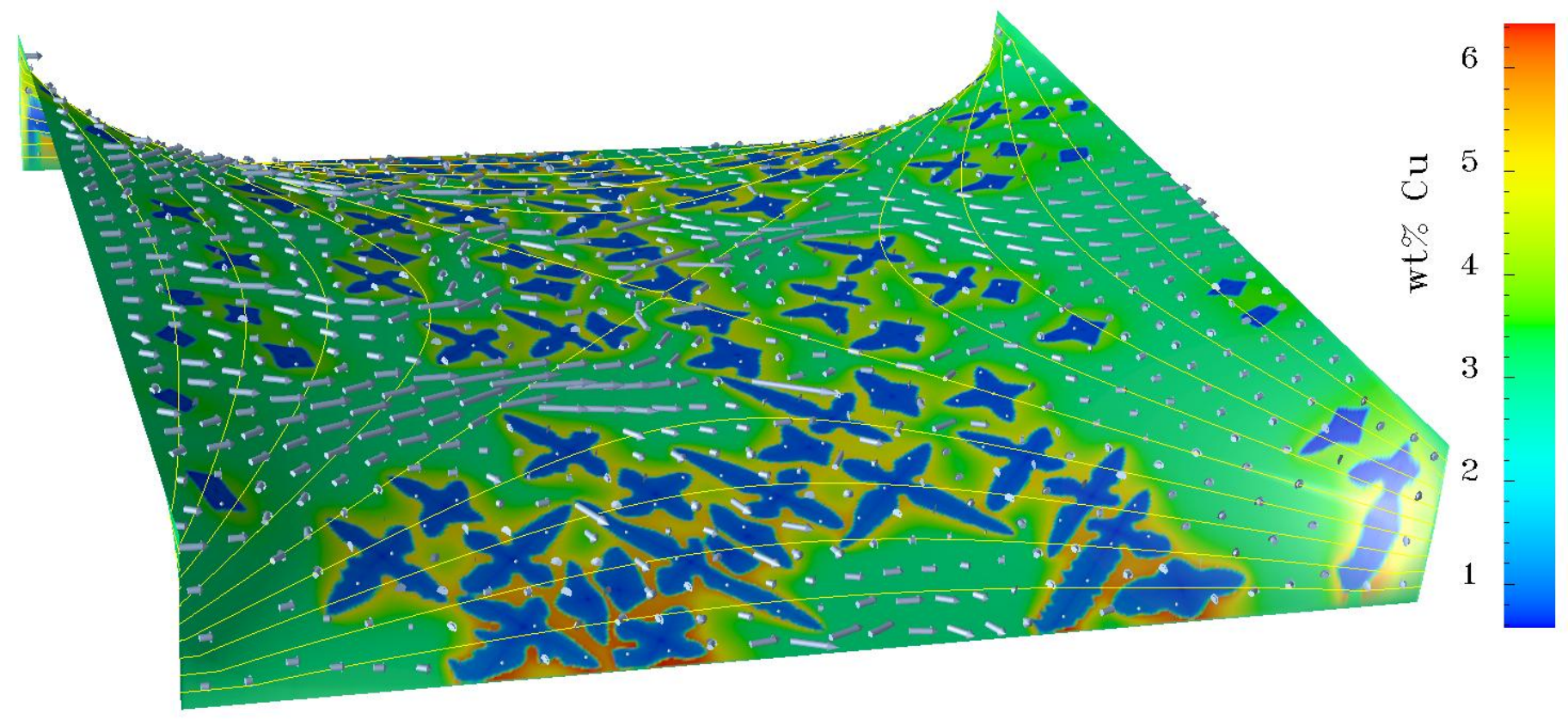


Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)

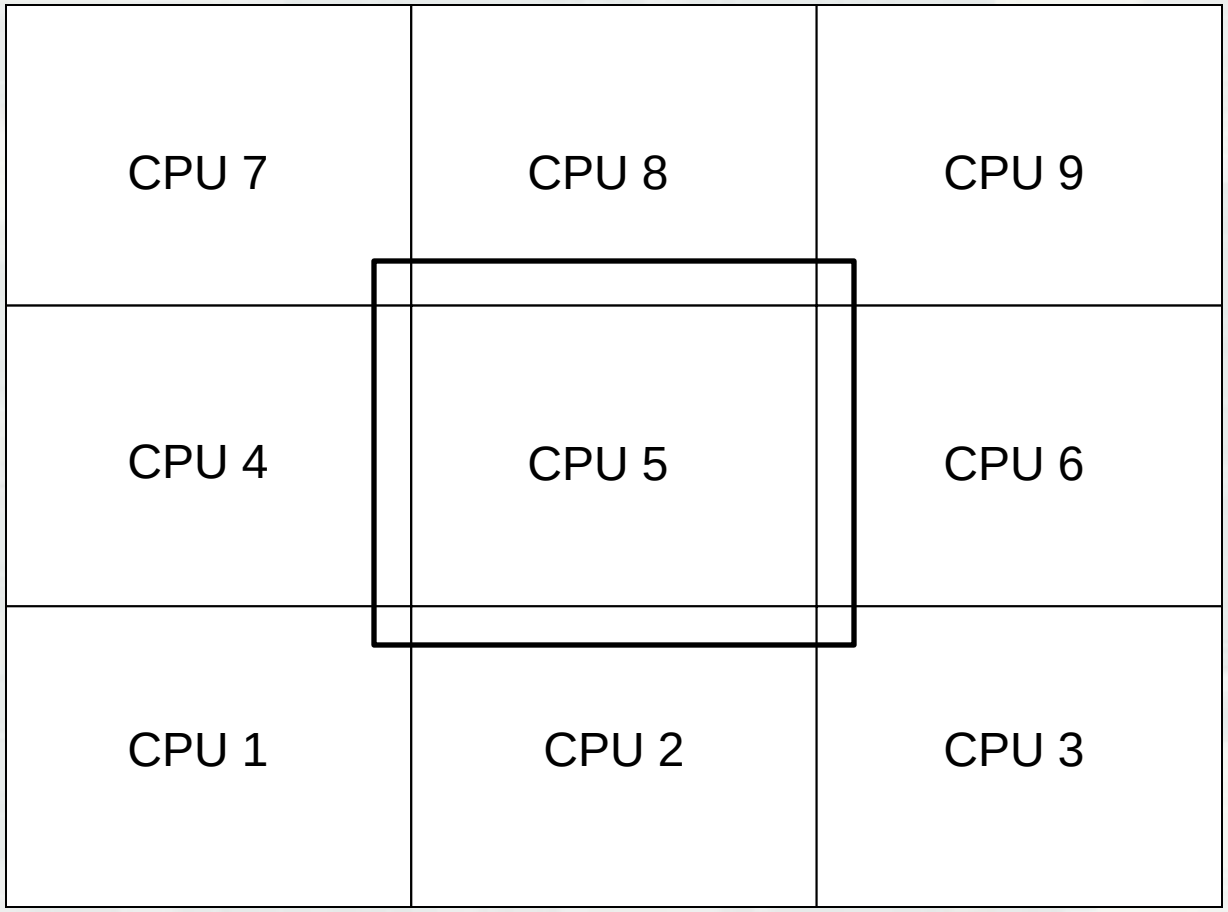
buffer=send
MPI_Sendrcv_replace(
  buffer, dst=3, src=7)
recv=buffer

LBM-CA solidification model C_l, u, T

Flow of solute between solidifying dendrites in a variable temperature field.
Cooled at front and back boundaries, heated from left (inlet) and right (outlet) boundaries.

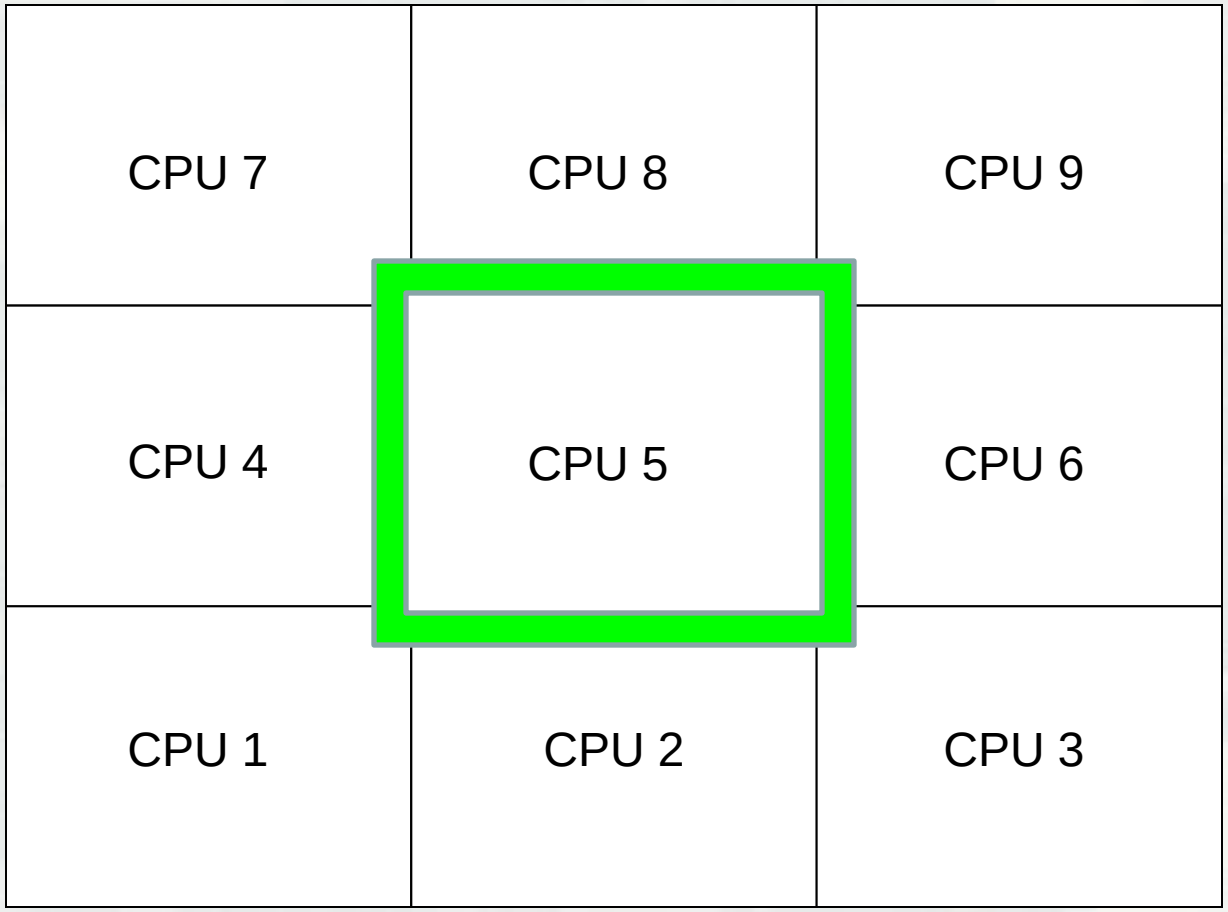


LBM-CA parallelization – dendrite growth



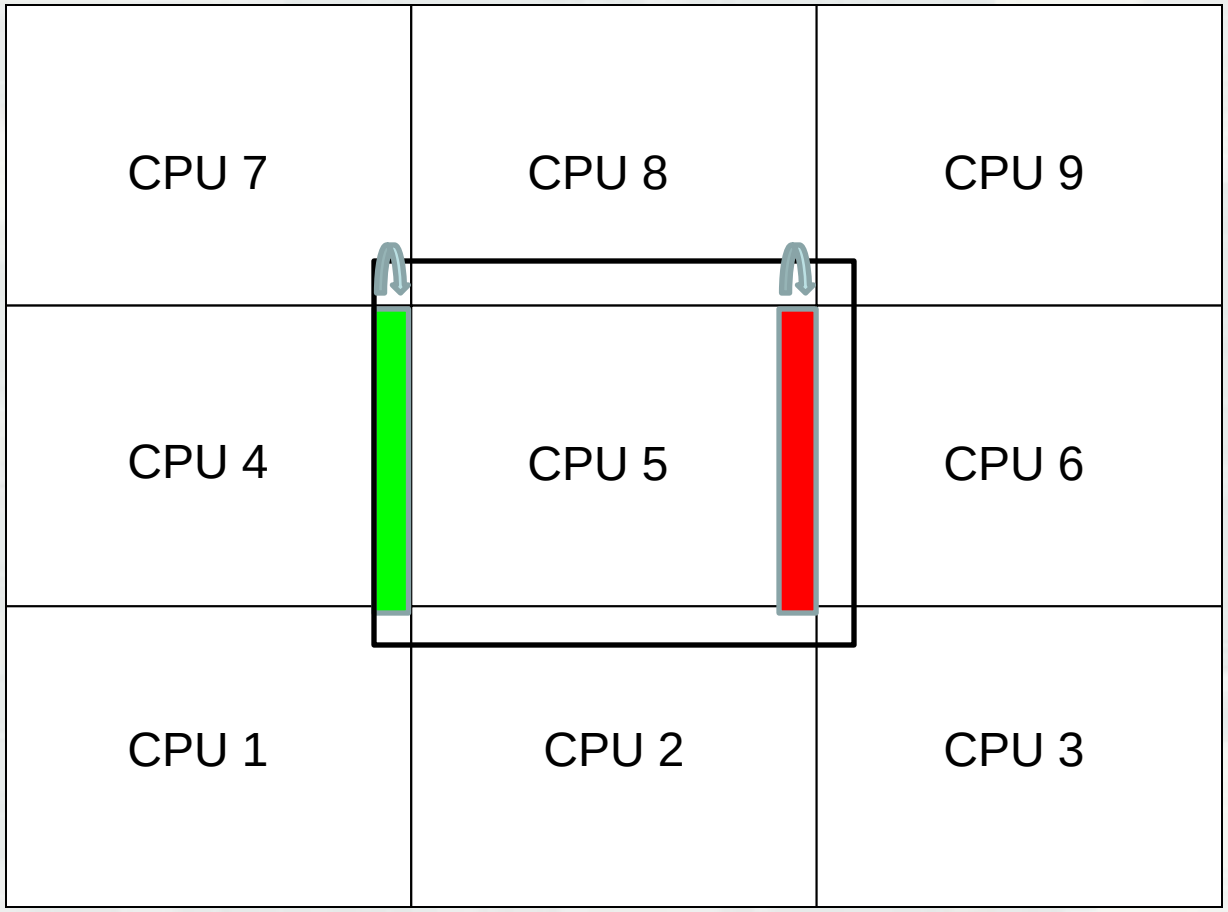
For dendrite growth, information from neighboring nodes is needed to update local node value

LBM-CA parallelization – ghost nodes



Populate ghost nodes after each local update

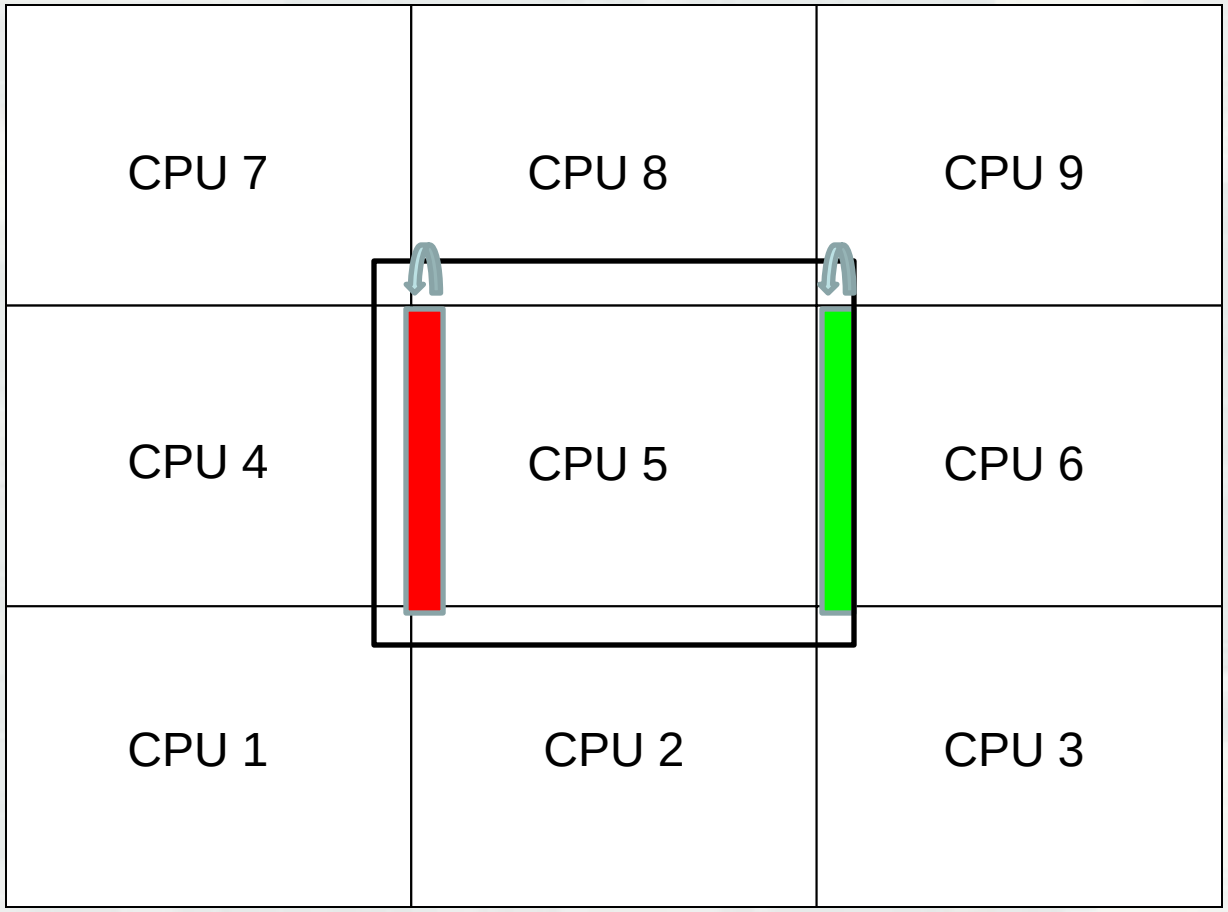
LBM-CA parallelization – ghost nodes



Populate ghost nodes after each local update east

MPI_Sendrcv(
send, recv,
dst=6, src=4)

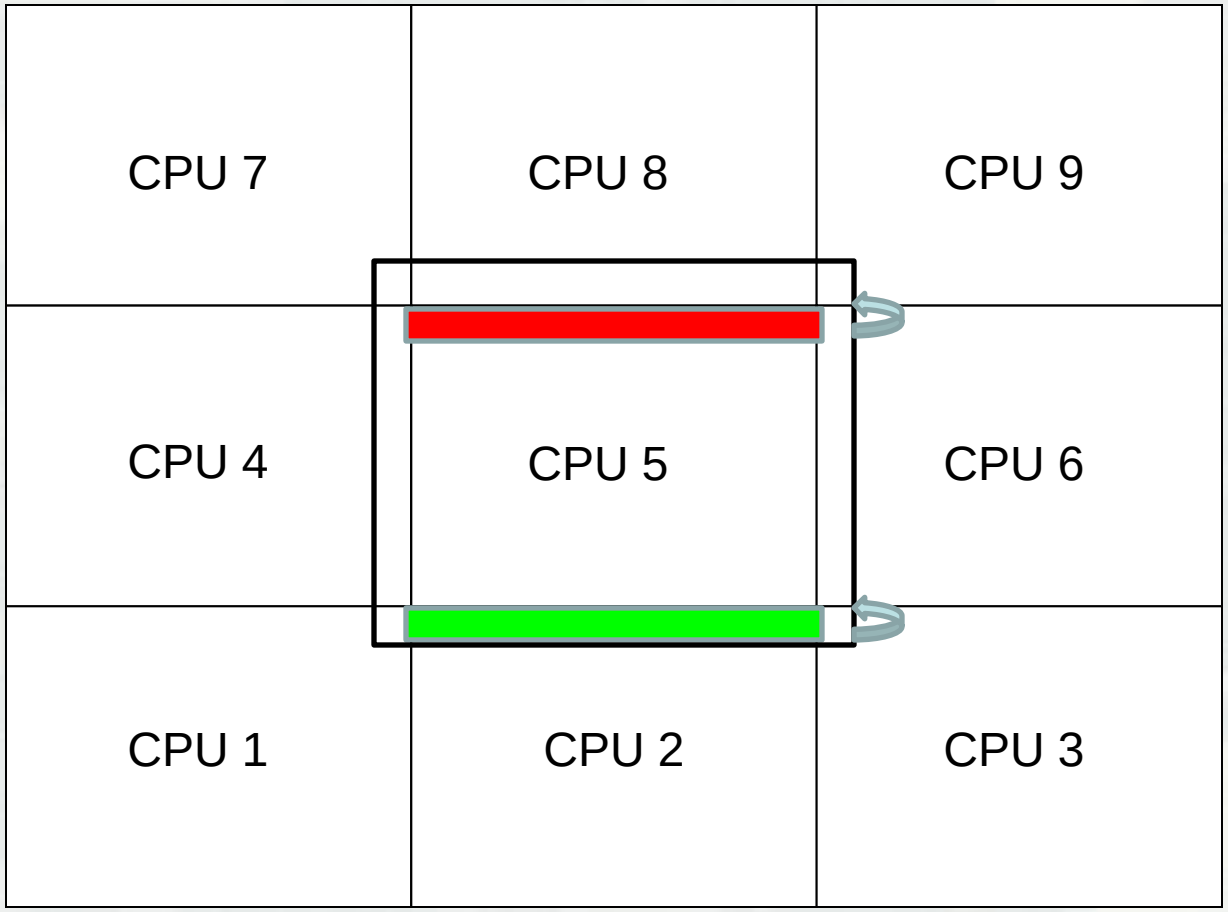
LBM-CA parallelization – ghost nodes



Populate ghost nodes after each local update west

MPI_Sendrcv(
send, recv,
dst=4, src=6)

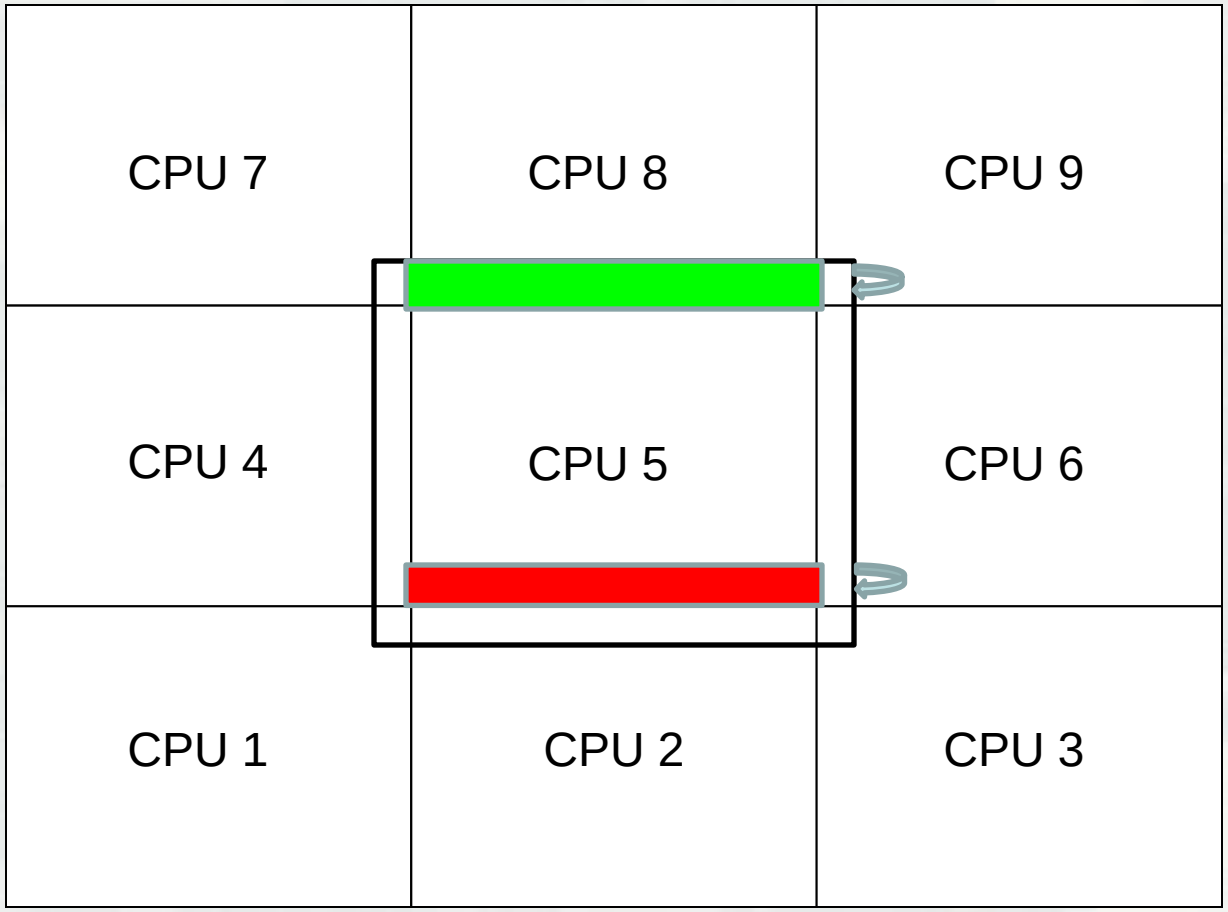
LBM-CA parallelization – ghost nodes



Populate ghost nodes after each local update north

MPI_Sendrcv(
send, recv,
dst=8, src=2)

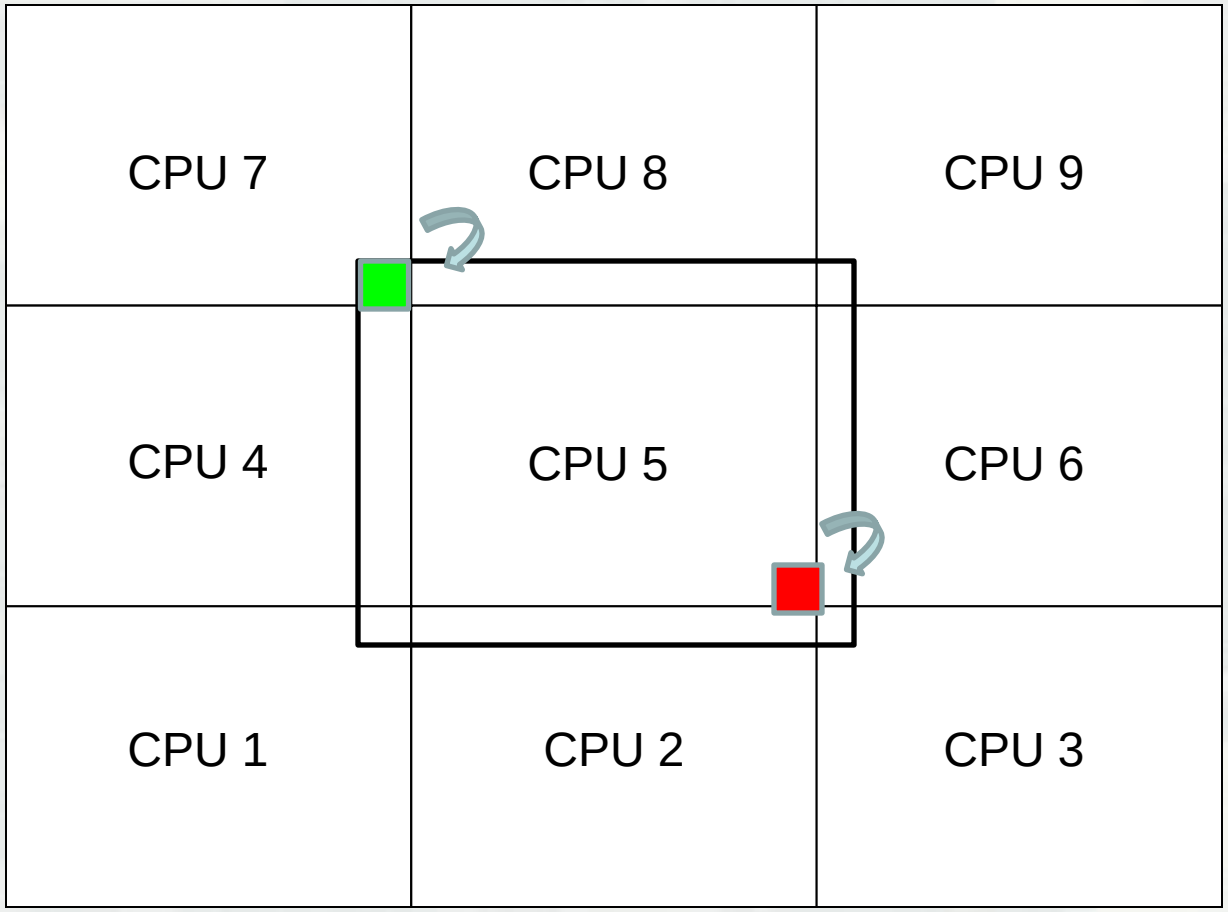
LBM-CA parallelization – ghost nodes



Populate ghost nodes after each local update south

MPI_Sendrcv(
send, recv,
dst=2, src=8)

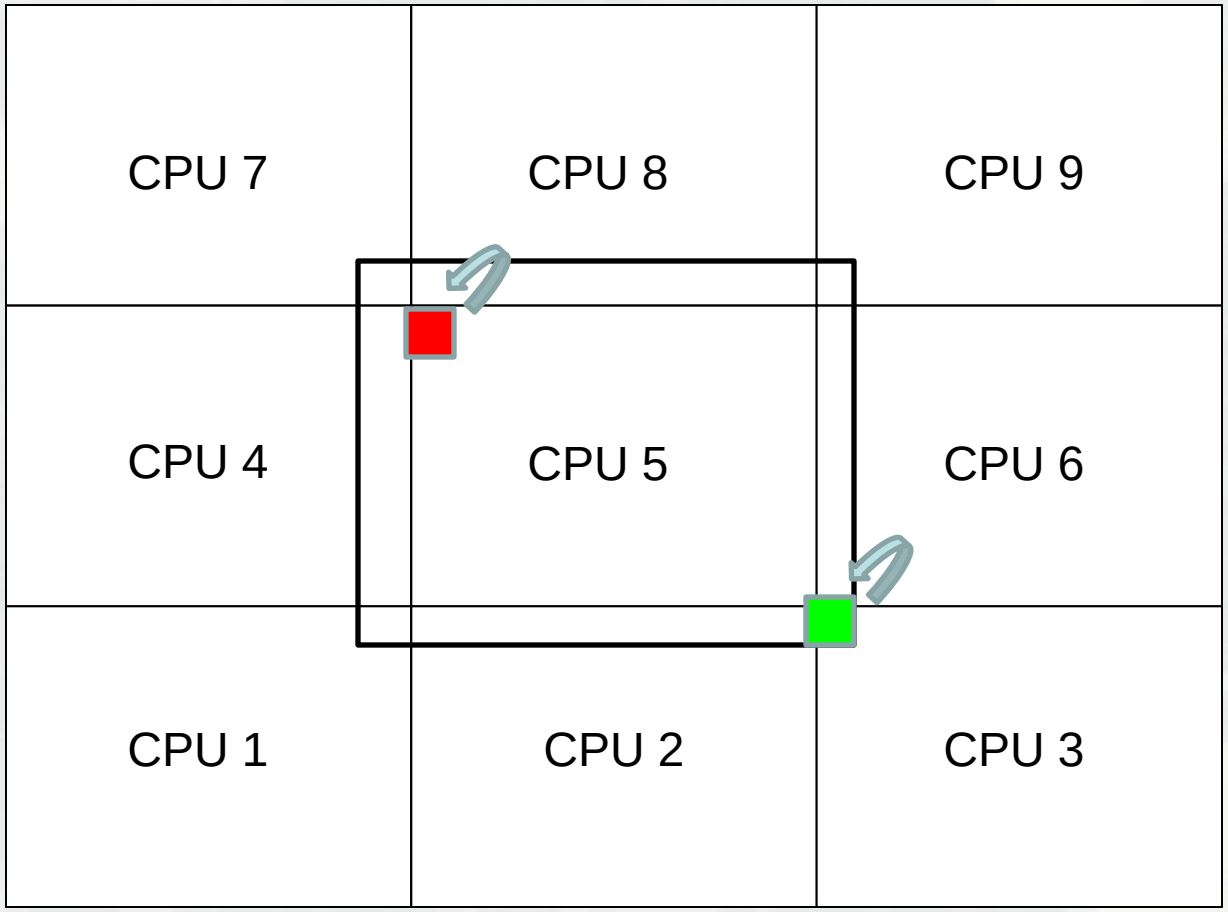
LBM-CA parallelization – ghost nodes



Populate ghost nodes after each local update south-east

MPI_Sendrcv(
send, recv,
dst=3, src=7)

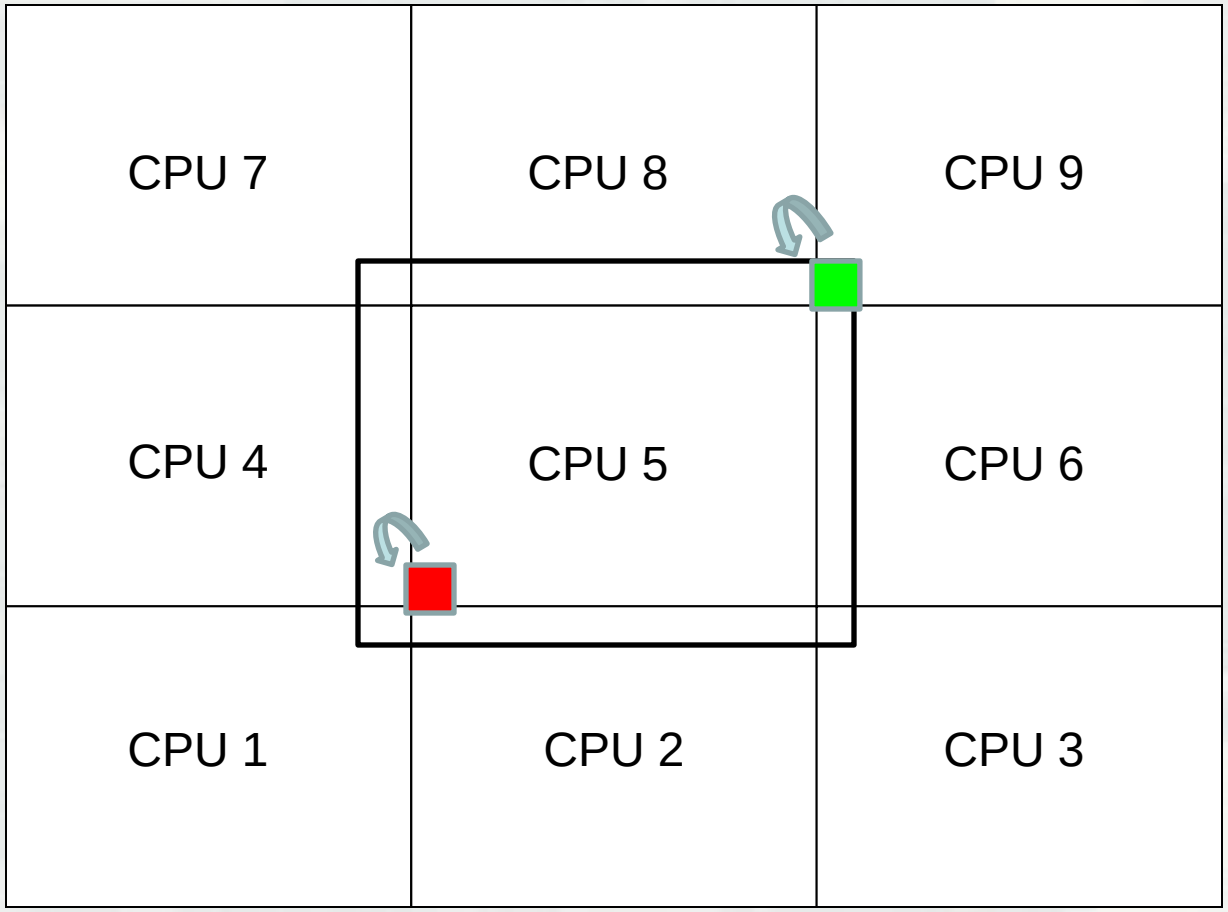
LBM-CA parallelization – ghost nodes



Populate ghost nodes after each local update north-west

MPI_Sendrcv(
send, recv,
dst=7, src=3)

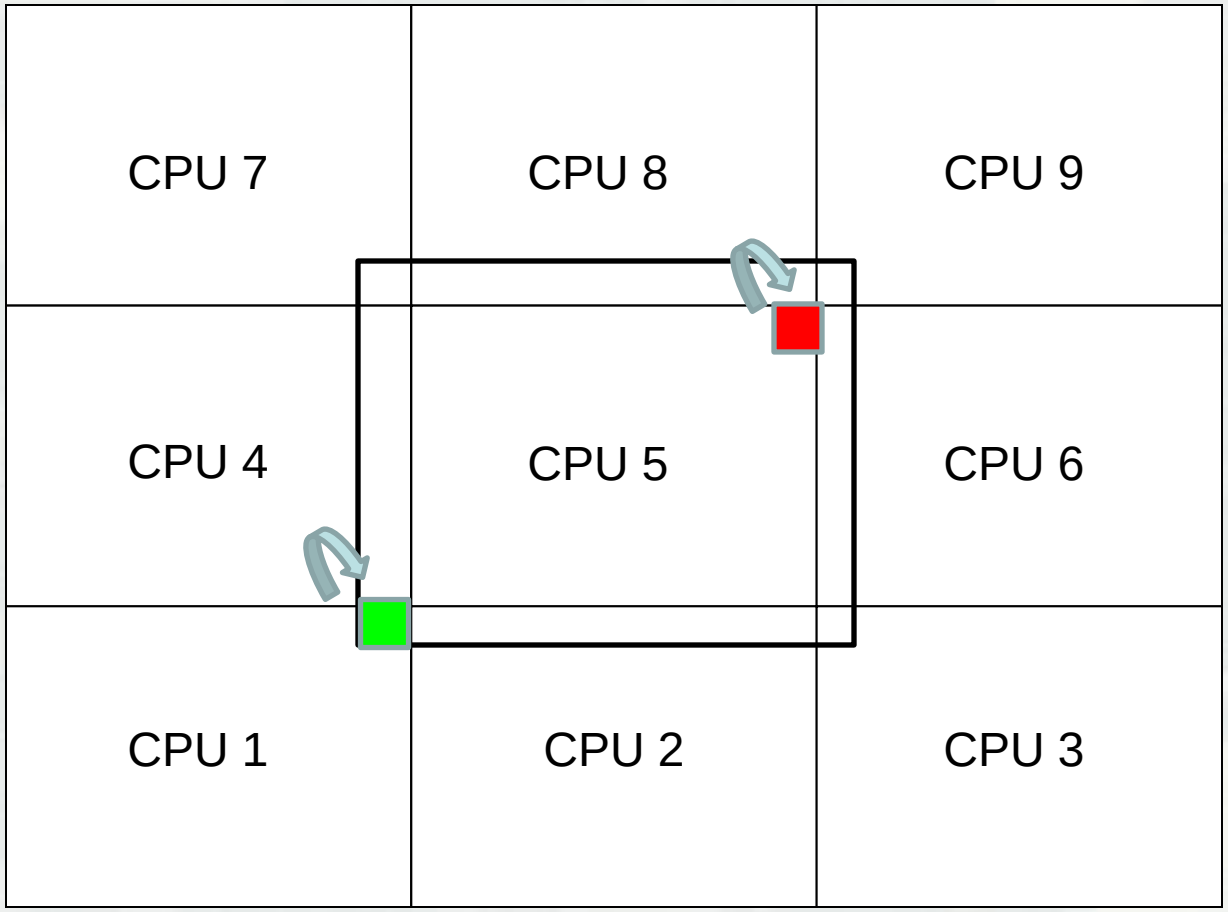
LBM-CA parallelization – ghost nodes



Populate ghost nodes after each local update south-west

MPI_Sendrcv(
send, recv,
dst=1, src=9)

LBM-CA parallelization – ghost nodes



Populate ghost nodes after each local update north-east

MPI_Sendrcv(
send, recv,
dst=9, src=1)

Computational resources

Talon, MSU HPC^2:

- 3072 cores, 12 cores/node (user limit 192 cores / job)
- Intel Xeon X5660 @2.8GHz (Westmere) processors
- 24 GByte/node memory
- Voltaire quad data-rate InfiniBand (40Gb/s)
- peak performance of over 34.4 TeraFLOPS

Kraken, NICS/ORNL:

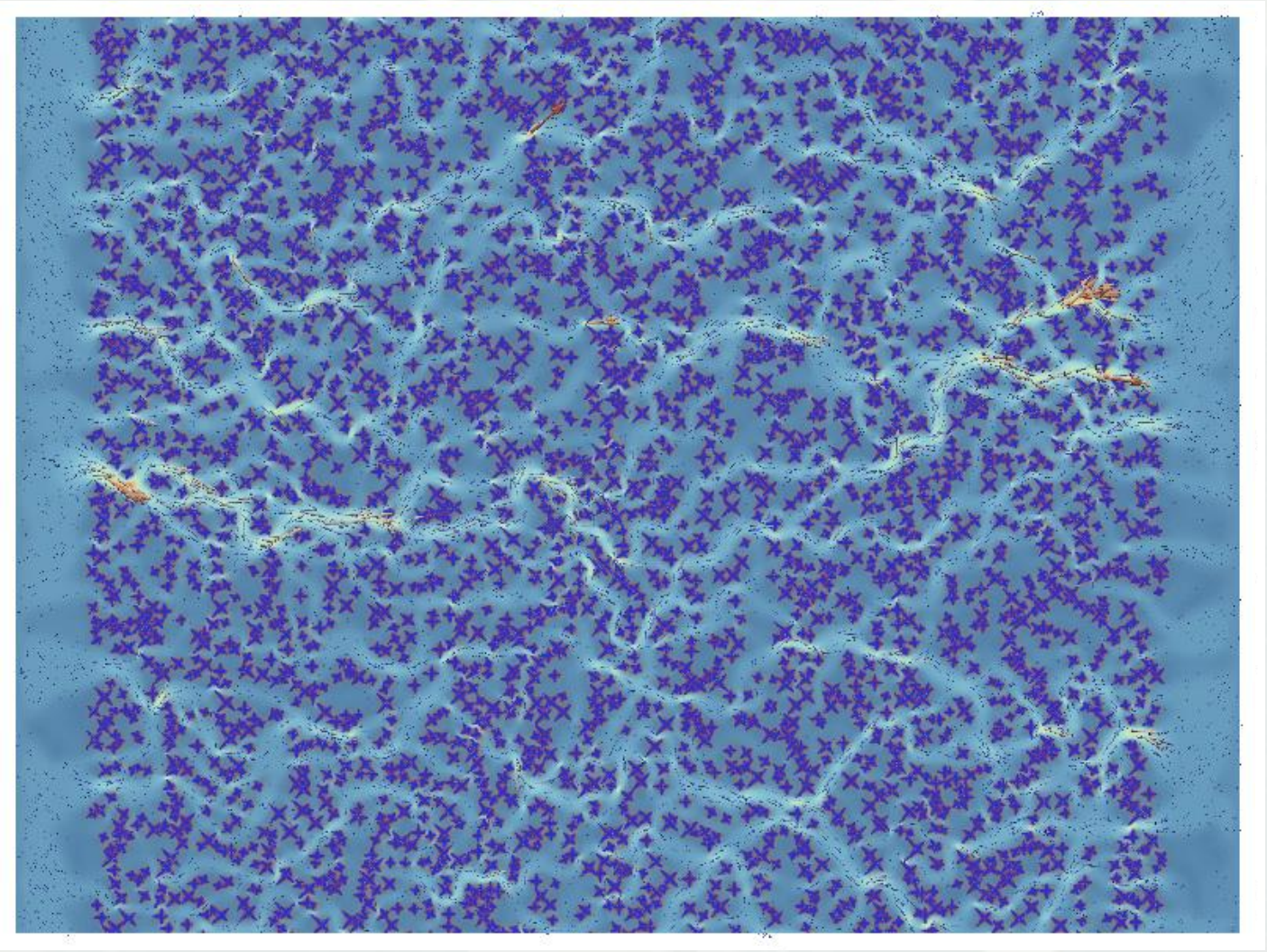
- 112,896 cores, 12 cores/node (user limit cores / job)
- AMD Opteron (Istanbul) @2.6GHz (Istanbul) processors
- 16 GByte/node memory
- Cray SeaStar2+ router
- peak performance of 1.17 PetaFLOPS

Generating an initial configuration for parallel scaling tests

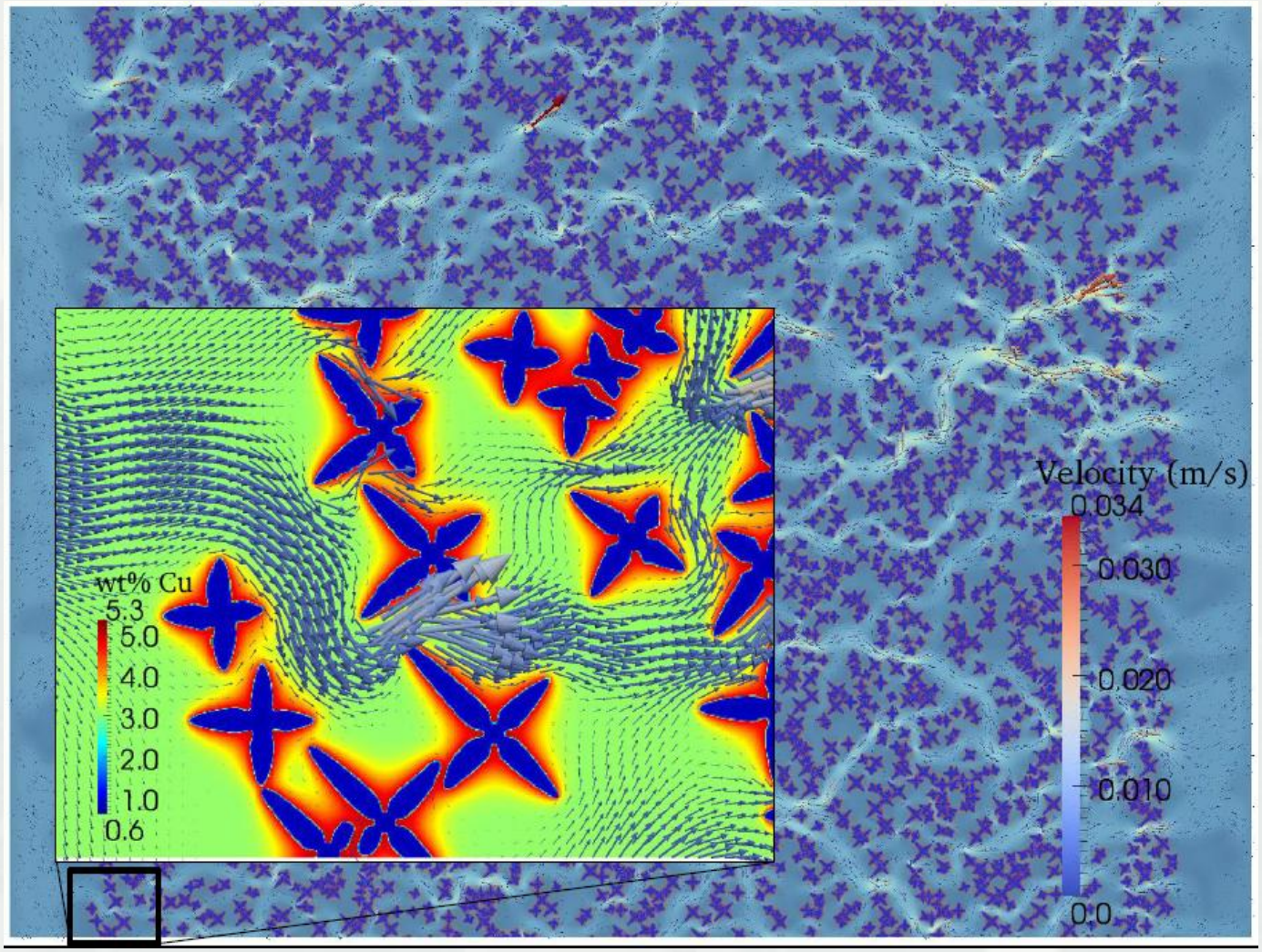
Simulation domain:

- rectangular lattice, 8000x6000 grid points
- dimensions: 2.4 mm x 1.8 mm (0.3 μm/lattice distance)
- 3264 random dendrite nucleation sites
- constant cooling rate 100K/s across the whole domain
- forced melt flow through inlet (left) and outlet (right) boundaries
- almost 16 GB of memory = single node of Kraken
- 400k time steps
- took about 10 hours on 192 cores on Talon @ MSU

Initial configuration



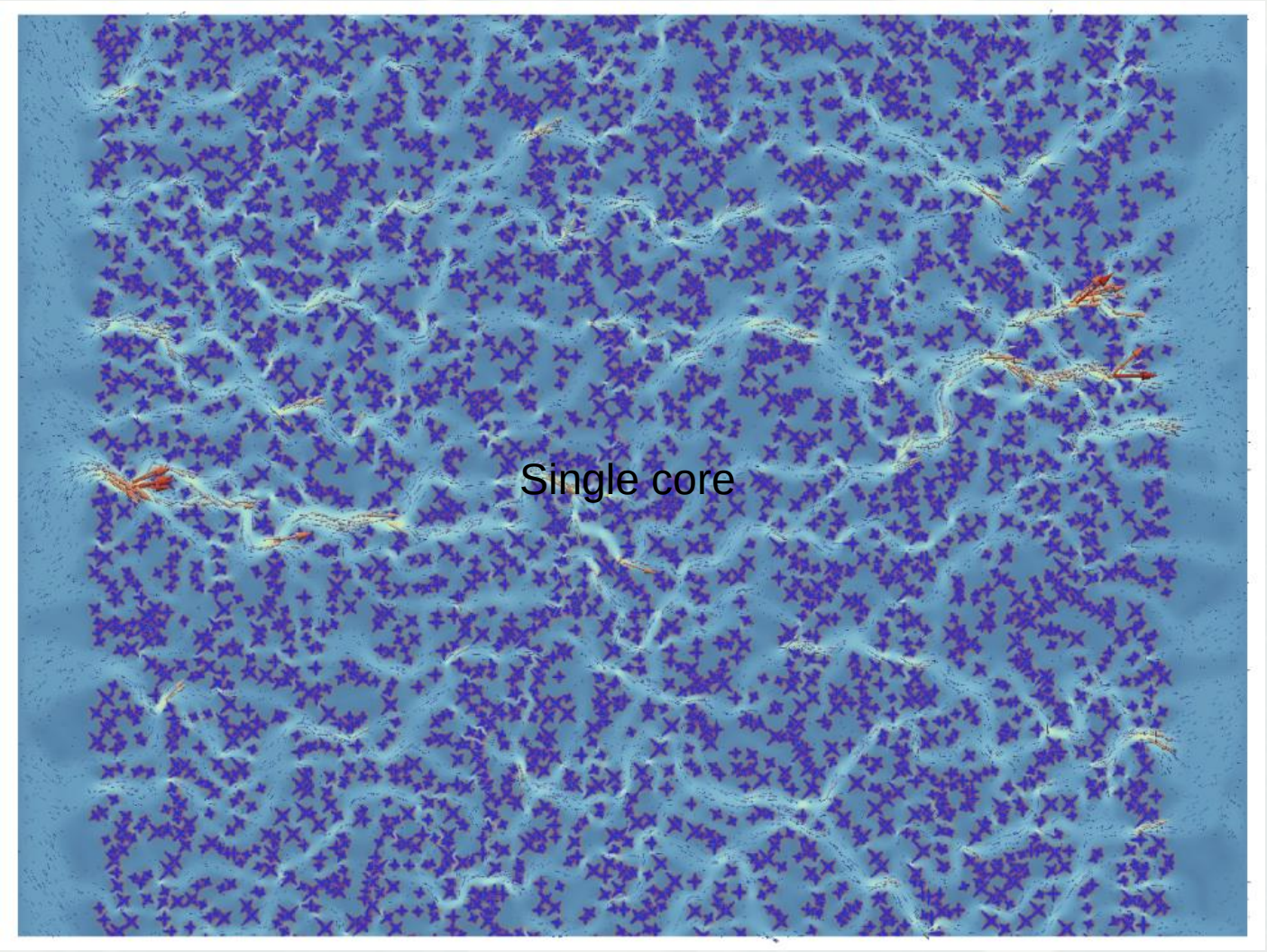
Magnified portion of initial configuration



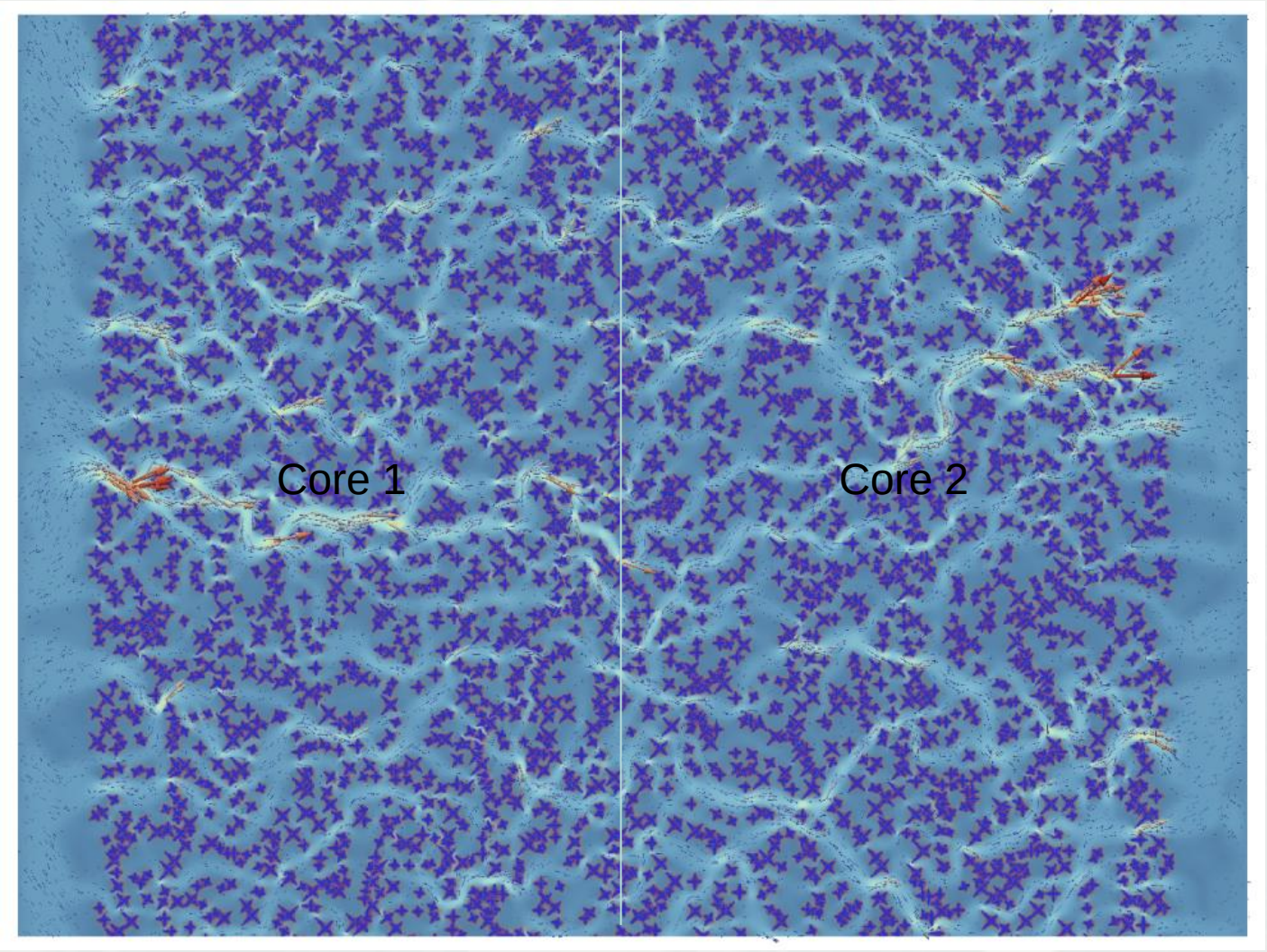
Speed up

- Speed up (strong scaling) represents how much faster a task is solved utilizing multiple cores
- Speed up tests were performed by restarting simulation from the step when the dendrites were fairly grown in the incubation domain
- Incubation domain is “split” equally between varying number of cores, then executed for 587 time steps with a flow forced at the inlet (left) and outlet (right), and with a specified cooling flow rate at all boundaries

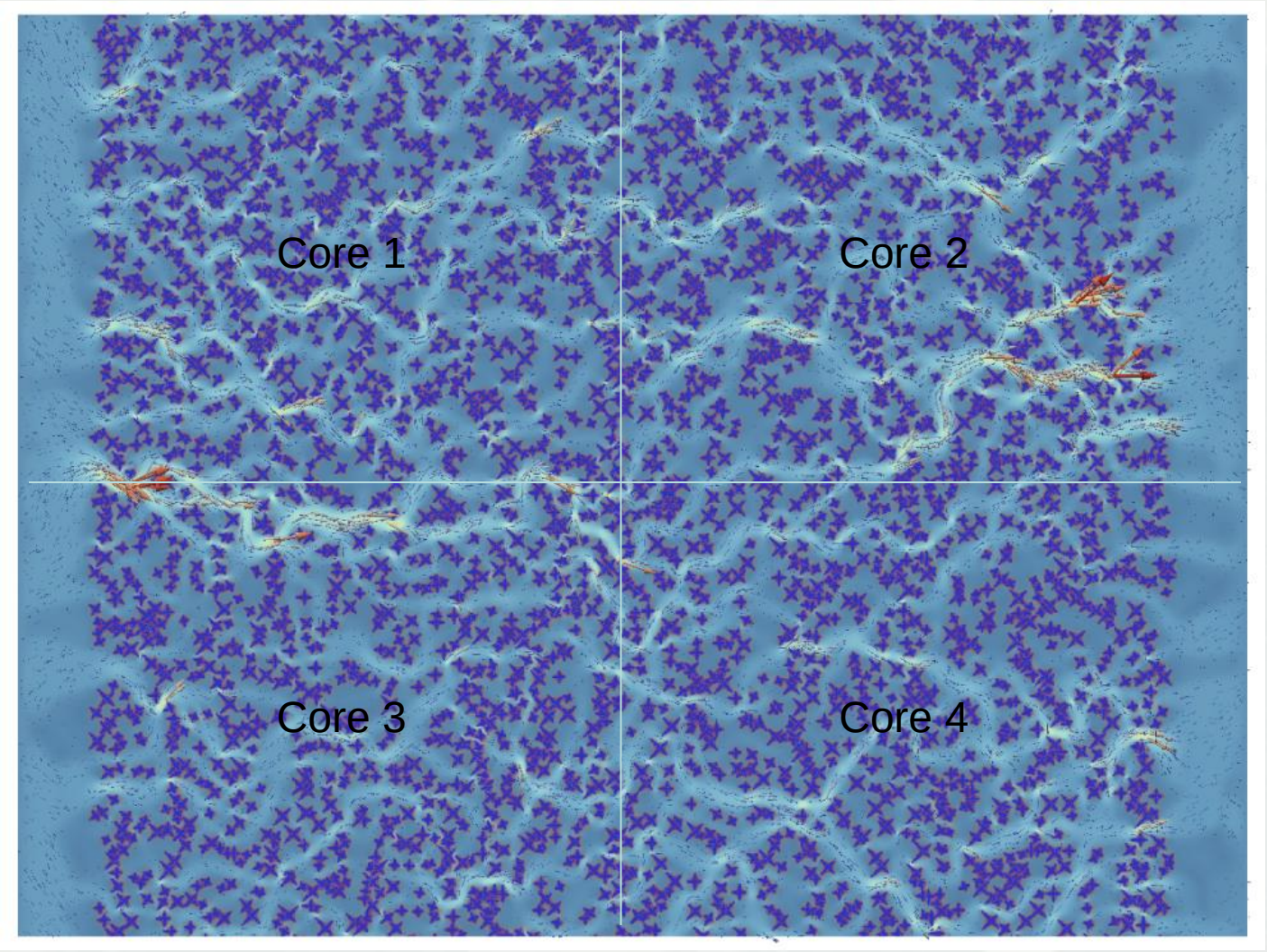
Speed up - constant task, 1 core



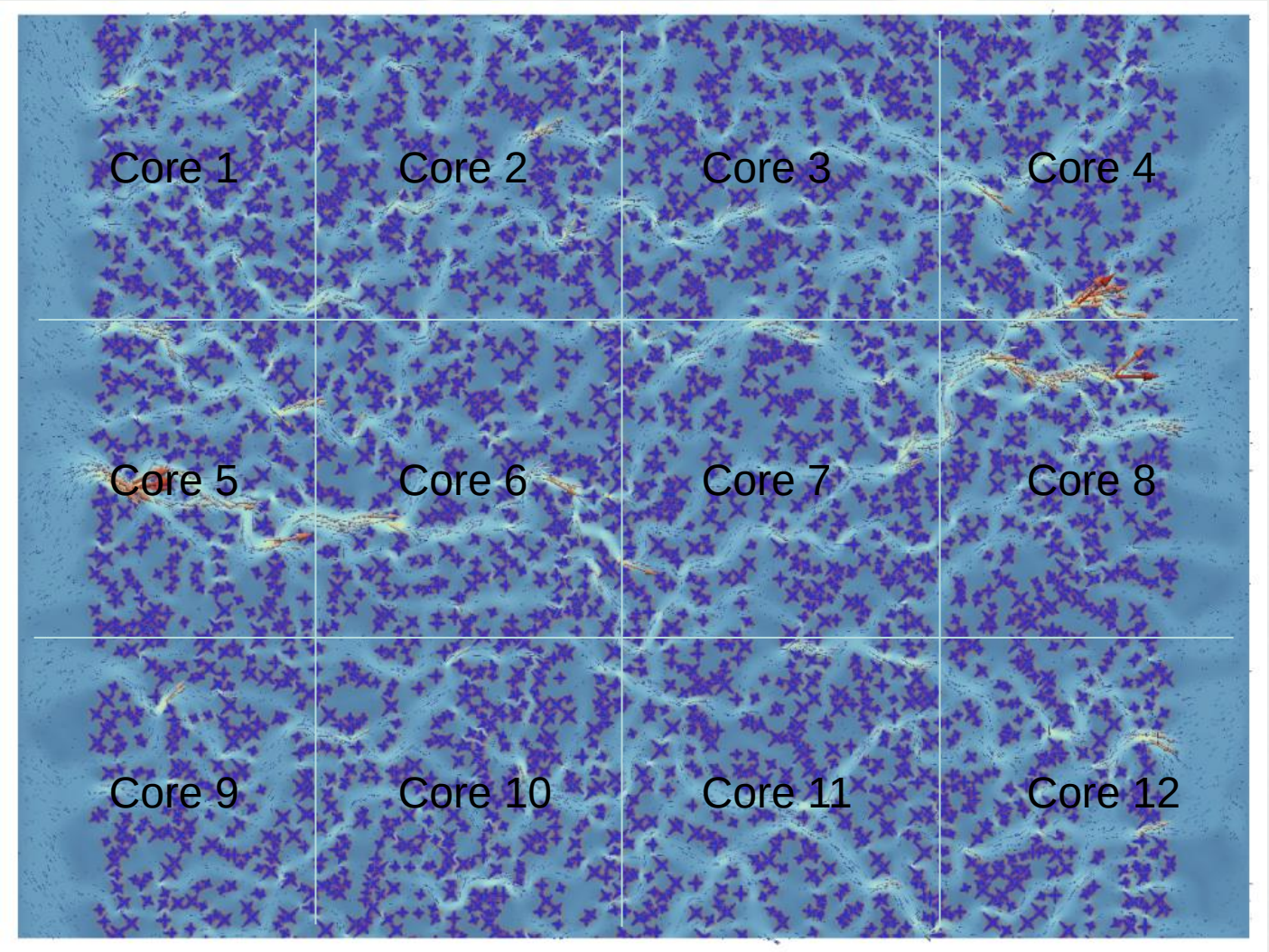
Speed up - constant task, 2 cores



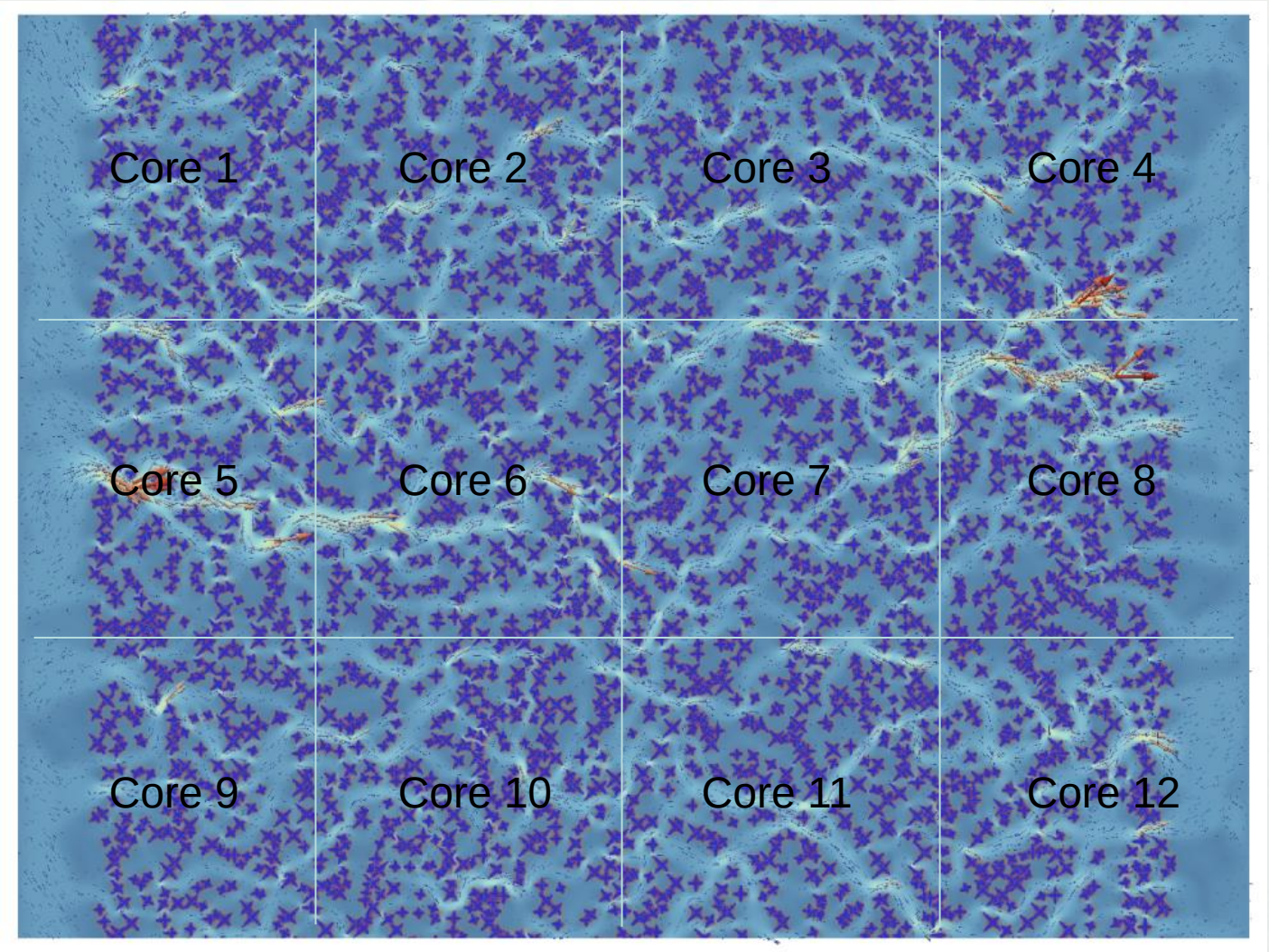
Speed up - constant task, 4 cores



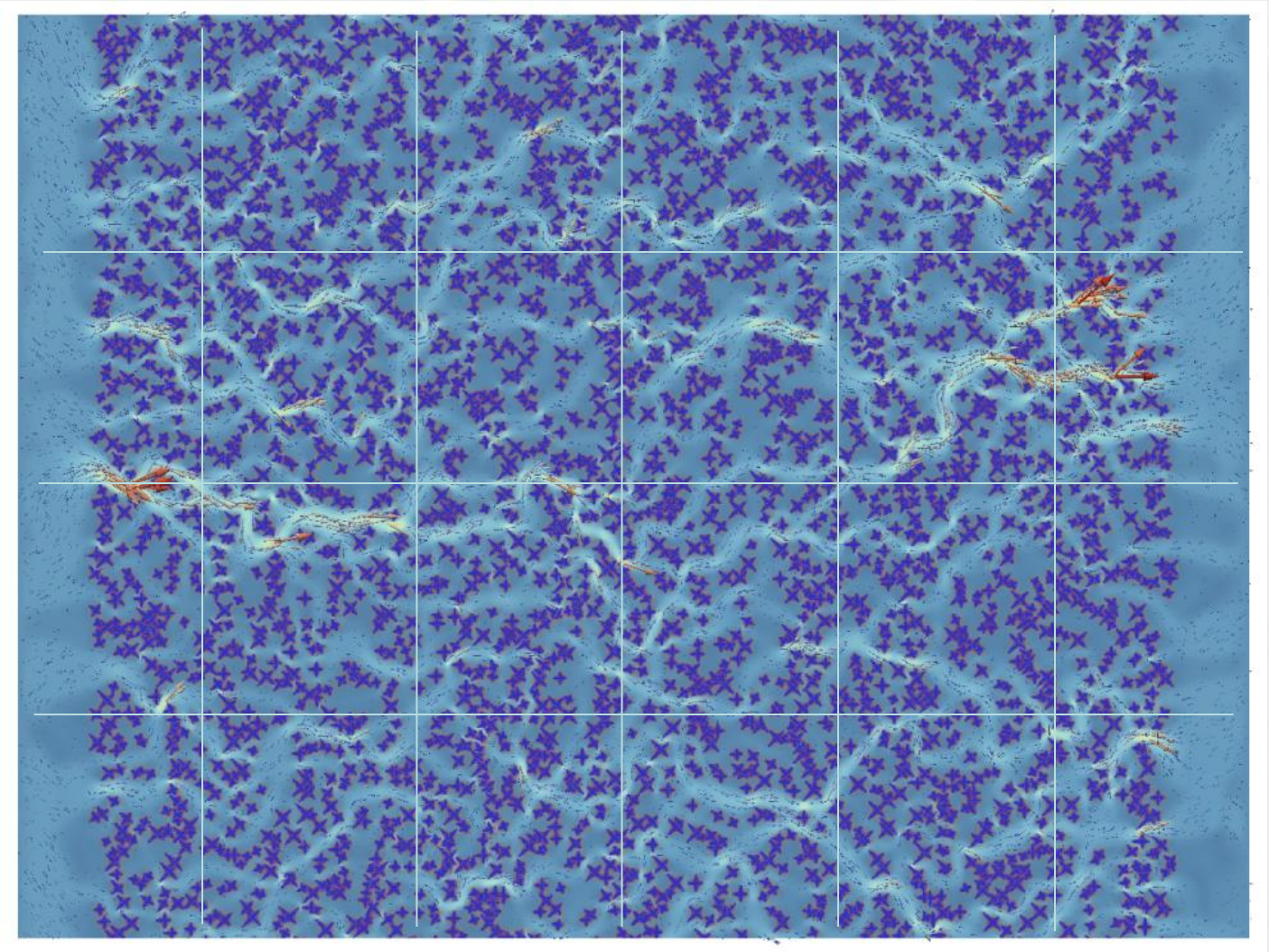
Speed up - constant task, 12 cores



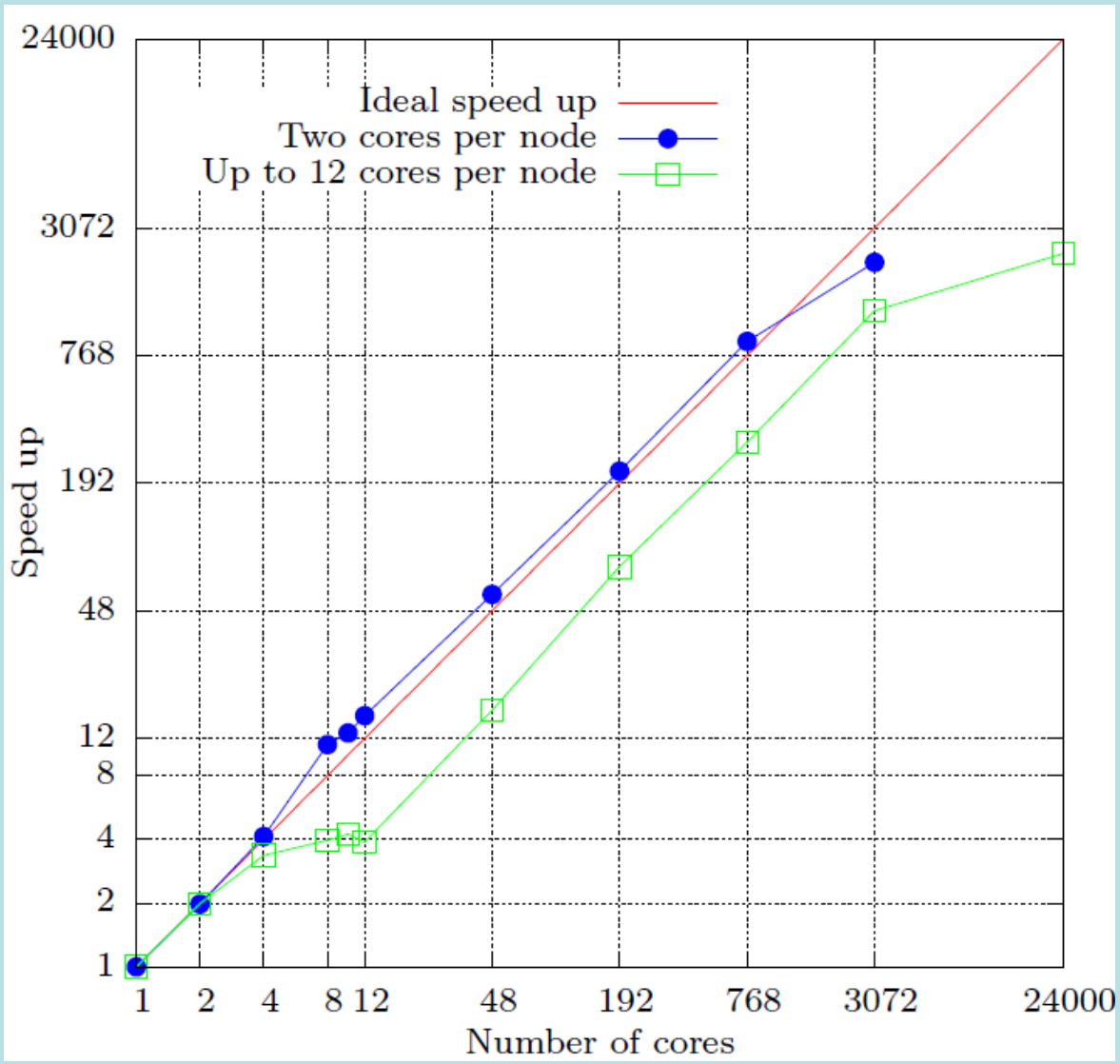
Speed up - constant task, 12 cores



Speed up - constant task, 24 cores



Speed up - results

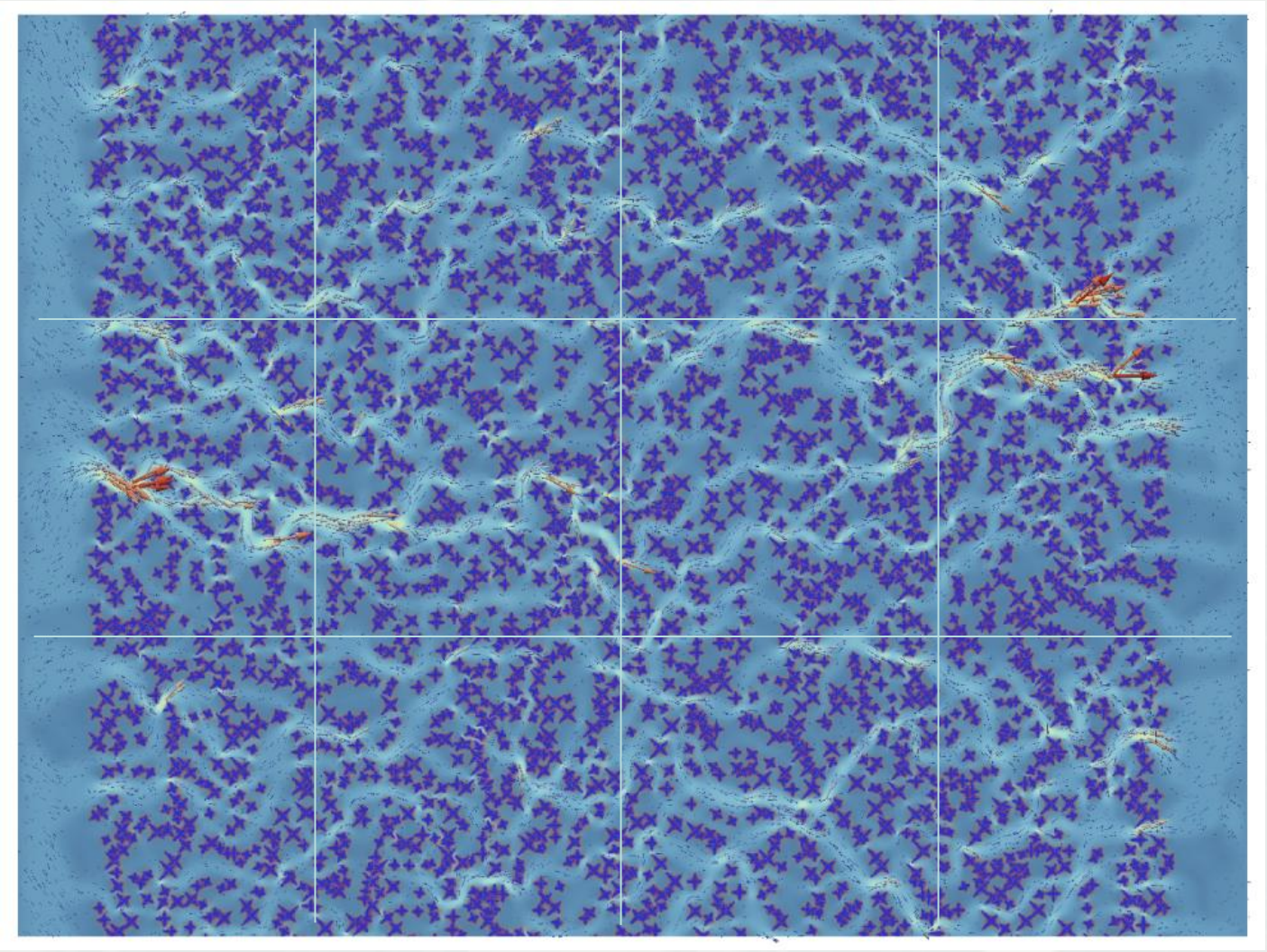


- strong scaling (speed up) near perfect up to 3072 cores
- Algorithm is memory bandwidth limited on multi-core architecture (low FLOP/byte ratio)

Scale up

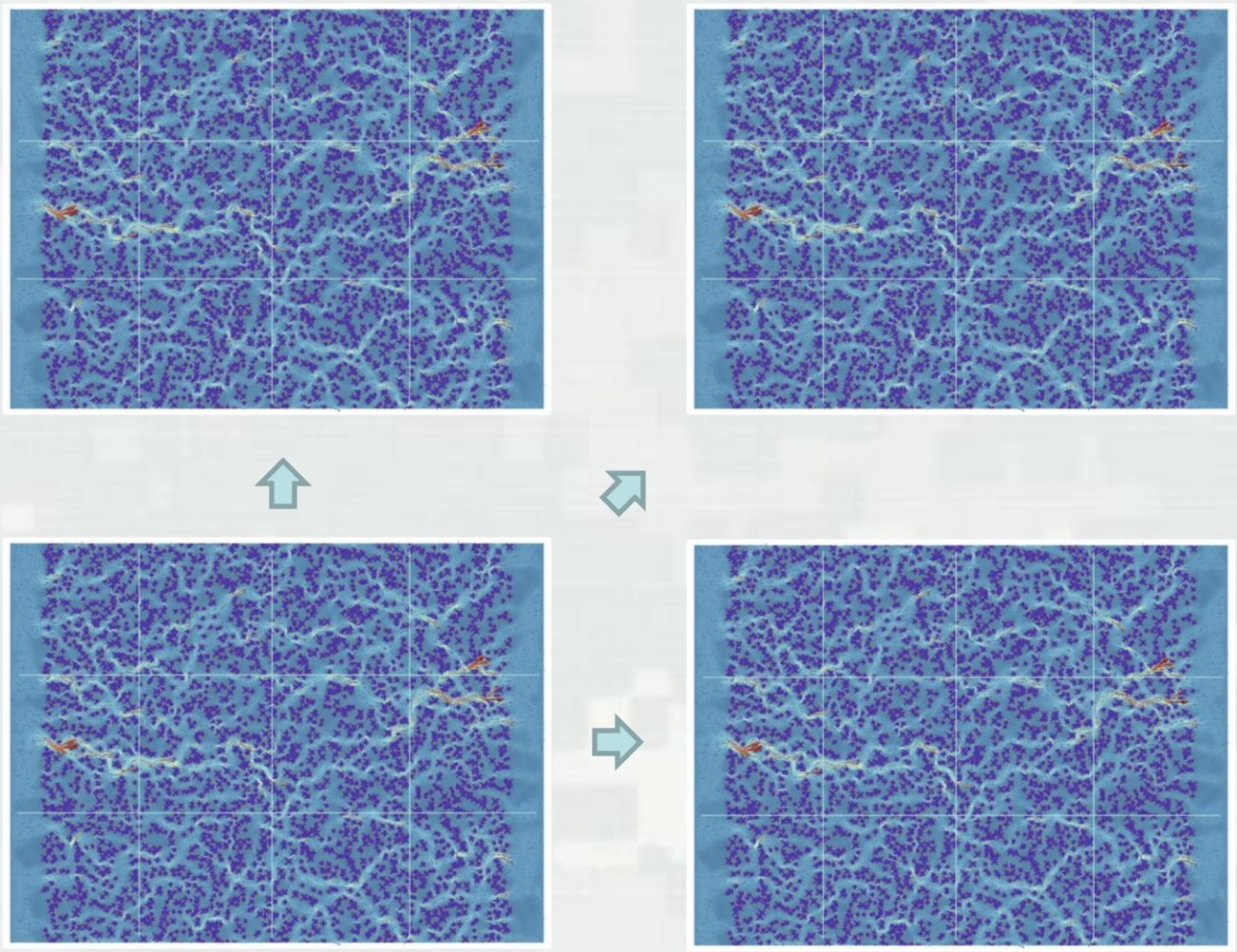
- Scale up (weak scaling) tests checks if the algorithm can solve larger task when more cores are utilized without a significant performance penalty
- Scale up tests were initialized from the stage when the dendrites were fairly grown in the incubation domain
- Incubated domain was “duplicated” equally onto varying number of nodes, then executed for 587 time steps with a flow forced at the inlet (left) and outlet (right), and with a specified cooling flow rate at all boundaries

Scale up - constant domain per node



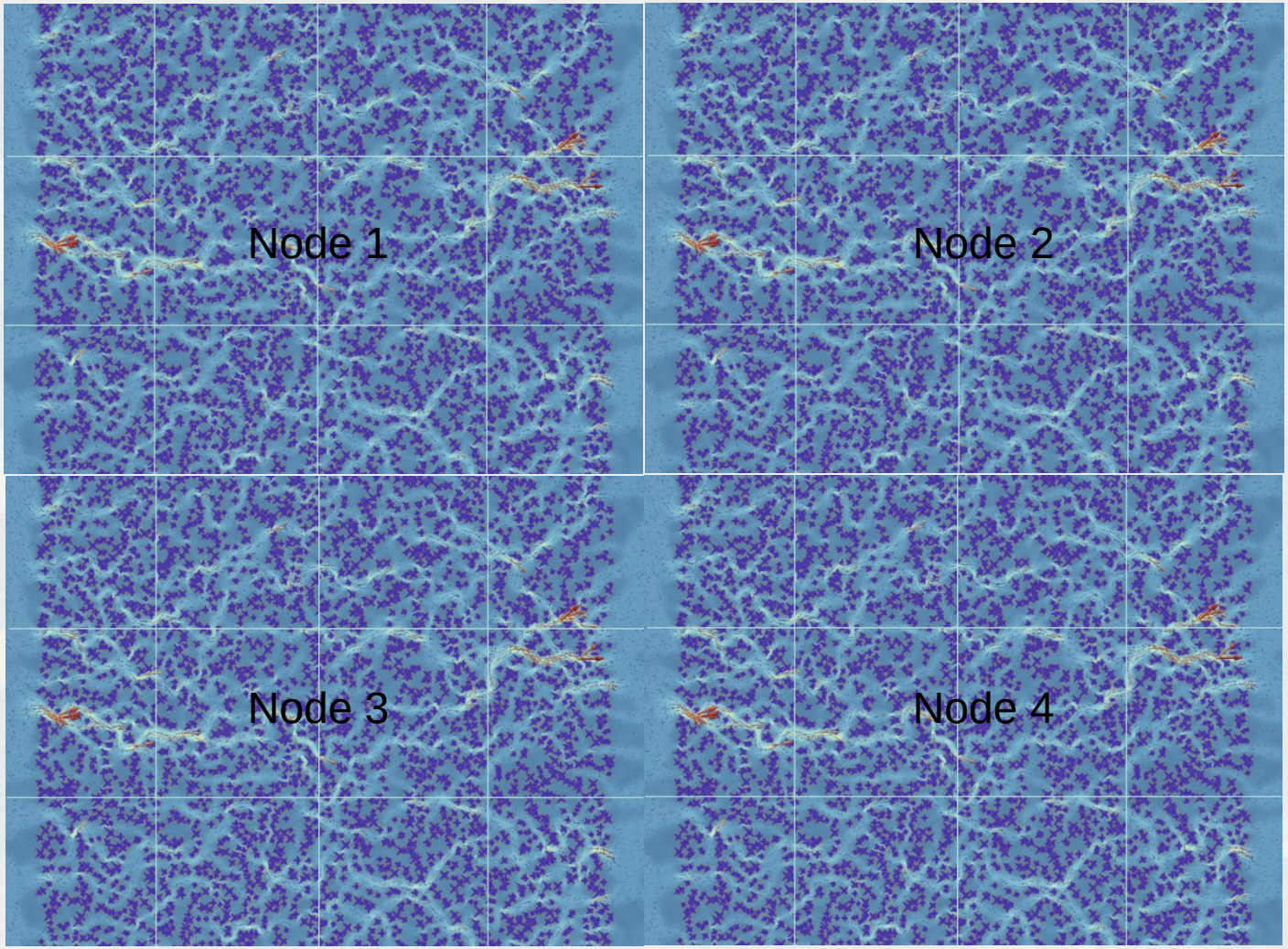
Base domain

Scale up - constant domain per node



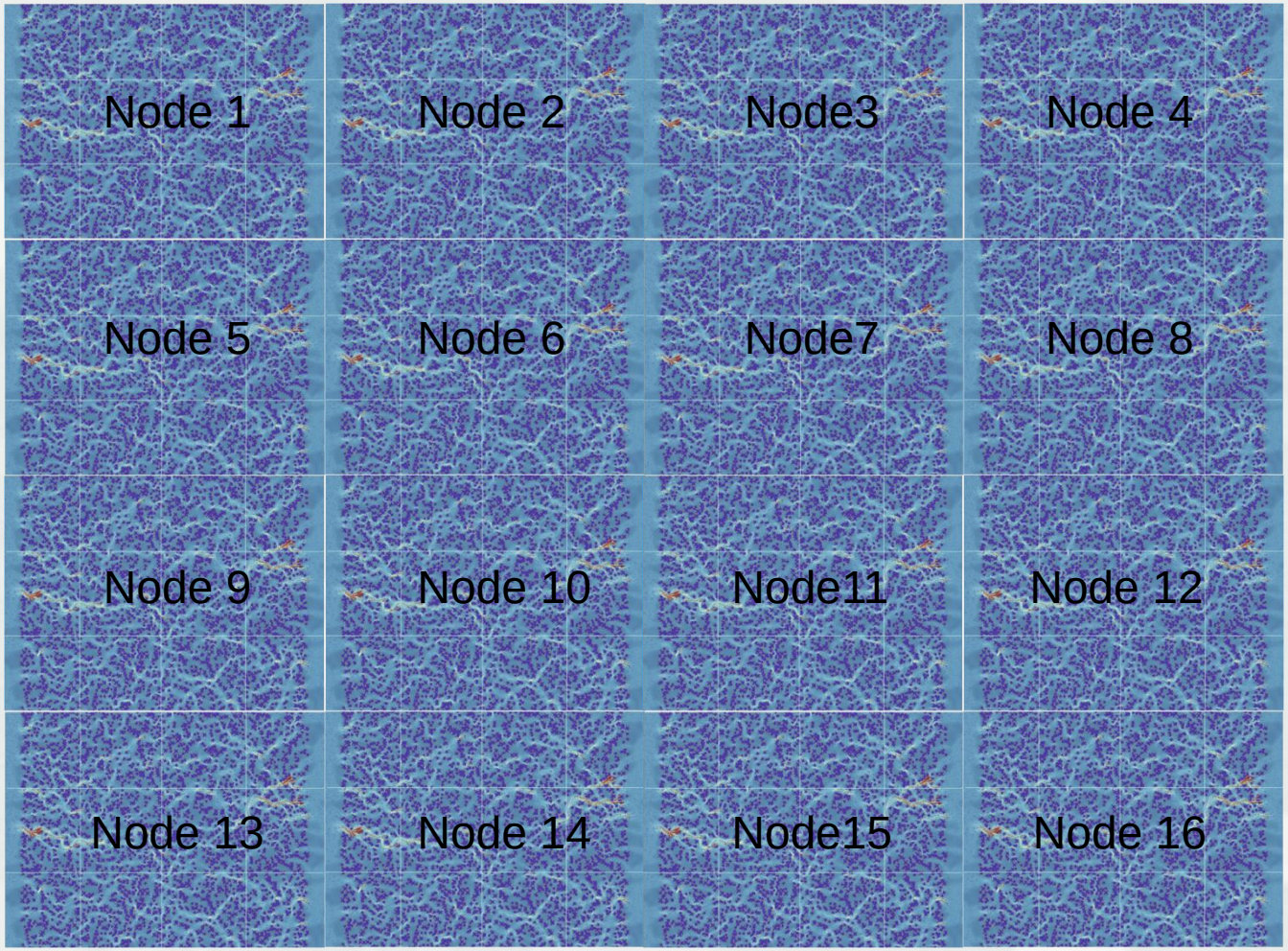
Duplication of the incubation domain onto 4 nodes

Scale up - constant domain per node



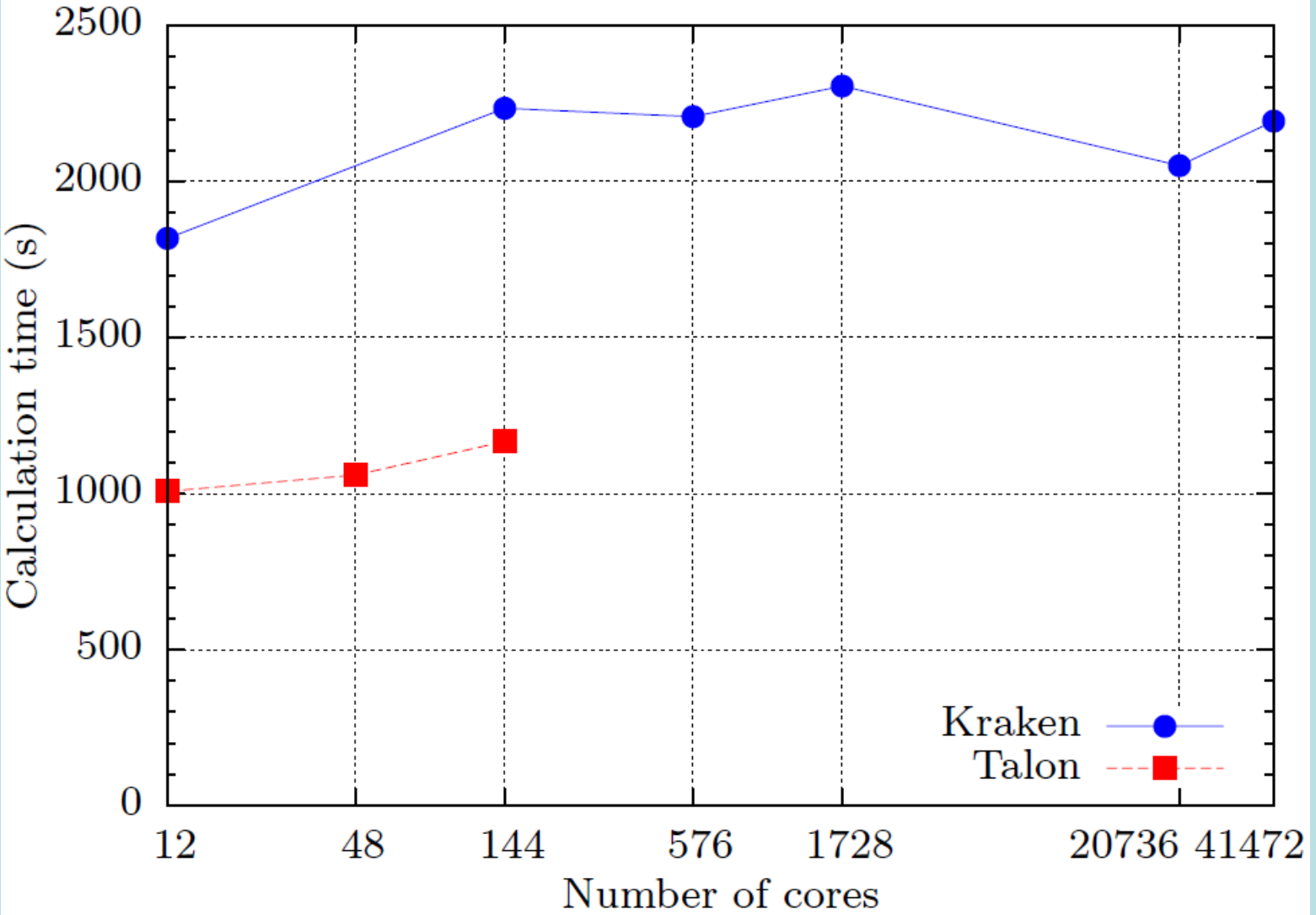
Duplication of the incubation domain onto 4 nodes

Scale up - constant domain per node



Duplication of the incubation domain onto 16 nodes

Scale up - results



Goal: constant calculation time

Scale up - results

Demonstrated nearly perfect scale up

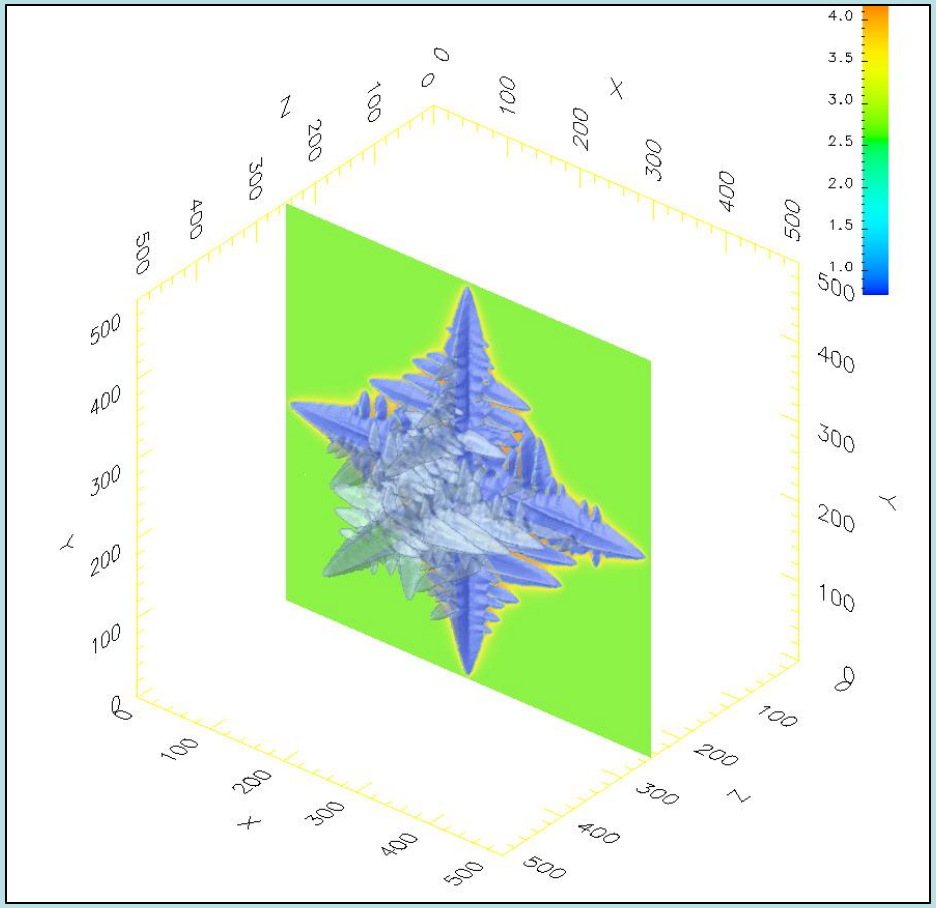
Largest domain:

- 41472 cores of Kraken
- over 165 billion grid nodes
- 11 millions of dendrites (only hundreds reported before)
- solute diffusion, melt convection, and heat transport
- dimensions 17.28 cm x 8.64 cm
- 587 time steps
- 40 minutes of simulation time

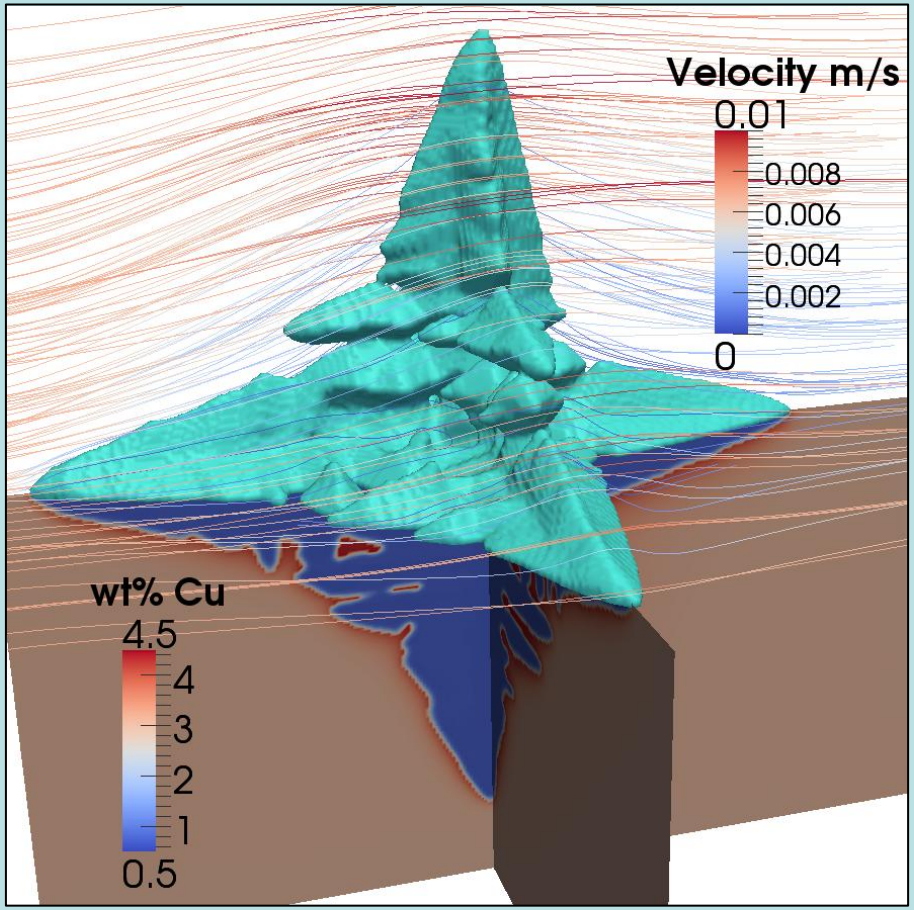
3D LBM-CA parallelization – C_l, u

3D Dendrite growth in undercooled Al-3wt%Cu melt

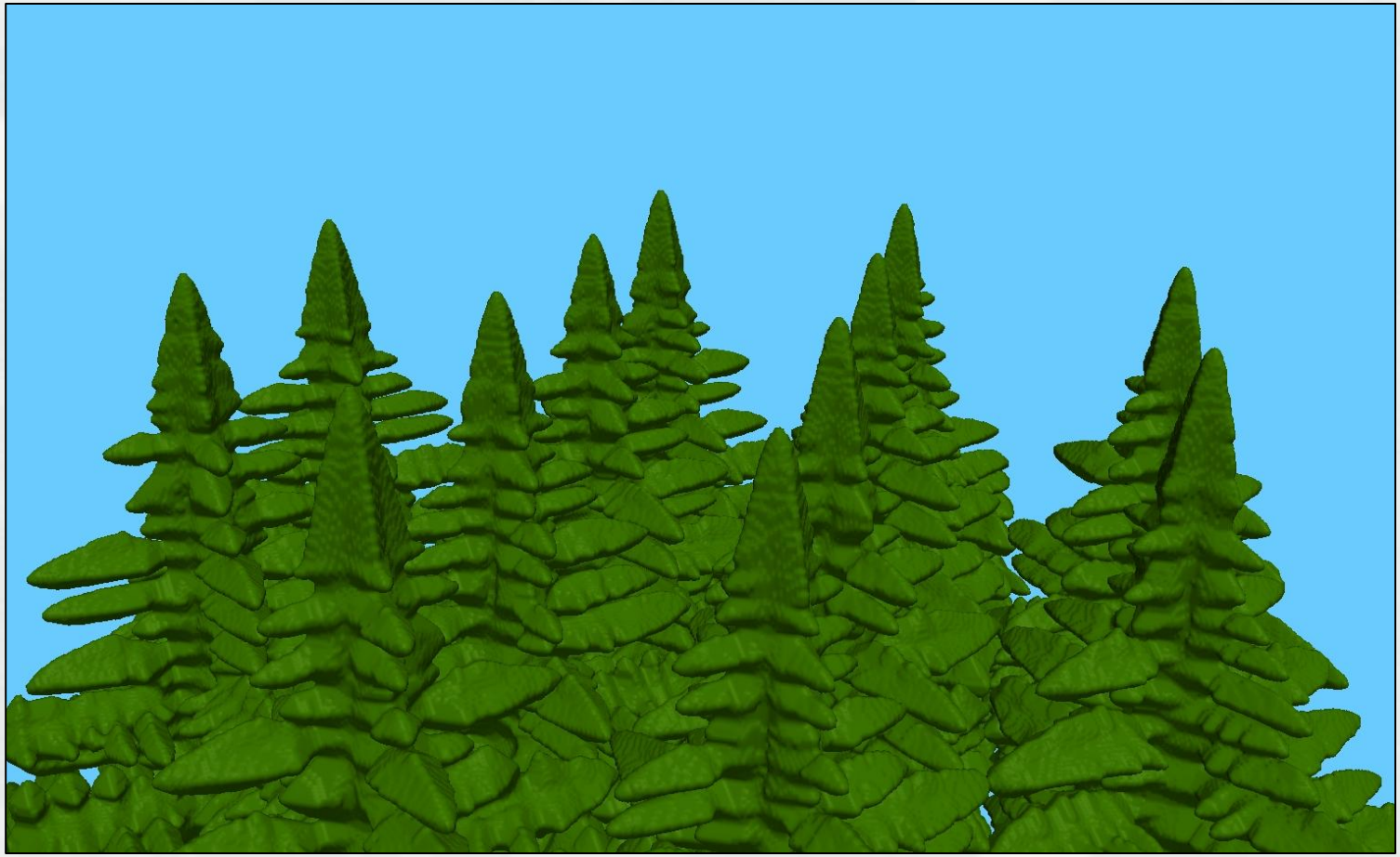
Equiaxial growth



Effect of convection



3D columnar dendrites growing in undercooled melt of Al-3wt%Cu



Domain size 180x180x144 $(\mu m)^3$

By Mohsen Eshraghi

Conclusions

2D:

- Parallelized 2D lattice Boltzmann / cellular automaton model of dendritic growth
- Tested the strong and weak parallel scaling of LBM/CA model with dendrites at advanced growth stage
- Demonstrated nearly ideal speed up and scale up

3D:

- Preliminary results exhibit similar speed up and scale up performance in 3D,
- measured tip growth velocity and solute concentration profiles
- Effects of convection, to be presented by Mohsen Eshraghi - Frontiers in Solidif. Science, Wed. 5:40 PM

Acknowledgement

Funding

- U.S. Army Corps ERDC
- NSF
- Center for Advanced Vehicular Systems @ MSU

Computational resources

- MSU HPC
- Kraken @ ORNL
- XSEDE collaborative support by Reuben Budiardja

Personal

- Mohsen Eshraghi for guiding me through the algorithm
- Sergio Felicelli and John Peters for opportunity to contribute to this interesting project