NANOSCALE FLUID FLOW

Bohumir Jelinek

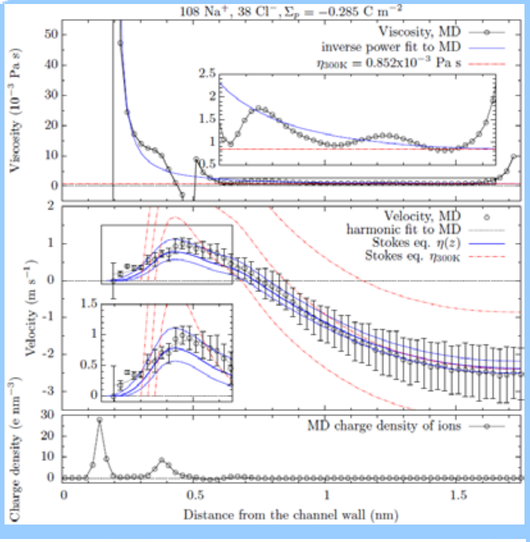
Postdoctoral Fellow

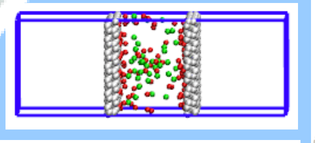
CAVS, Mississippi State University

Sergio Felicelli

Professor, Mechanical Engineering

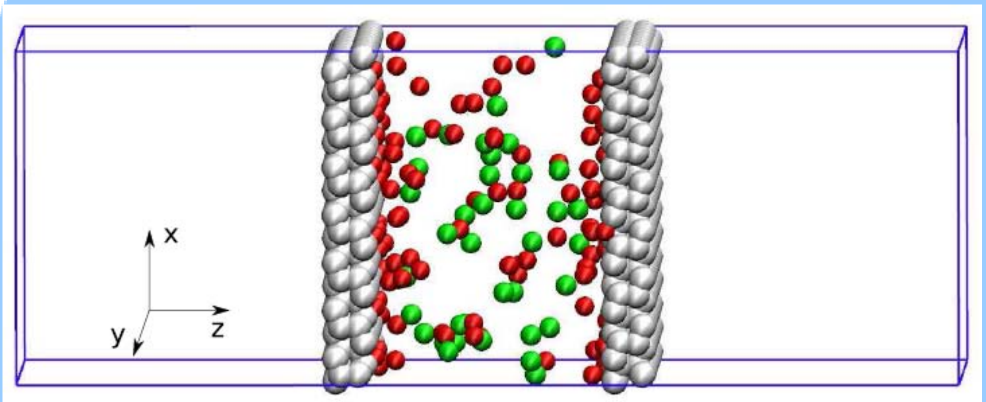
CAVS, Mississippi State University





$$\frac{d}{dz}\left[\eta(z)\frac{du_x(z)}{dz}\right] = -F_d(z)$$

$$\mathbf{F}_d(z) = e\left[c_{\mathrm{Na}^+}(z) - c_{\mathrm{Cl}^-}(z)\right]\mathbf{E}_{ext}$$

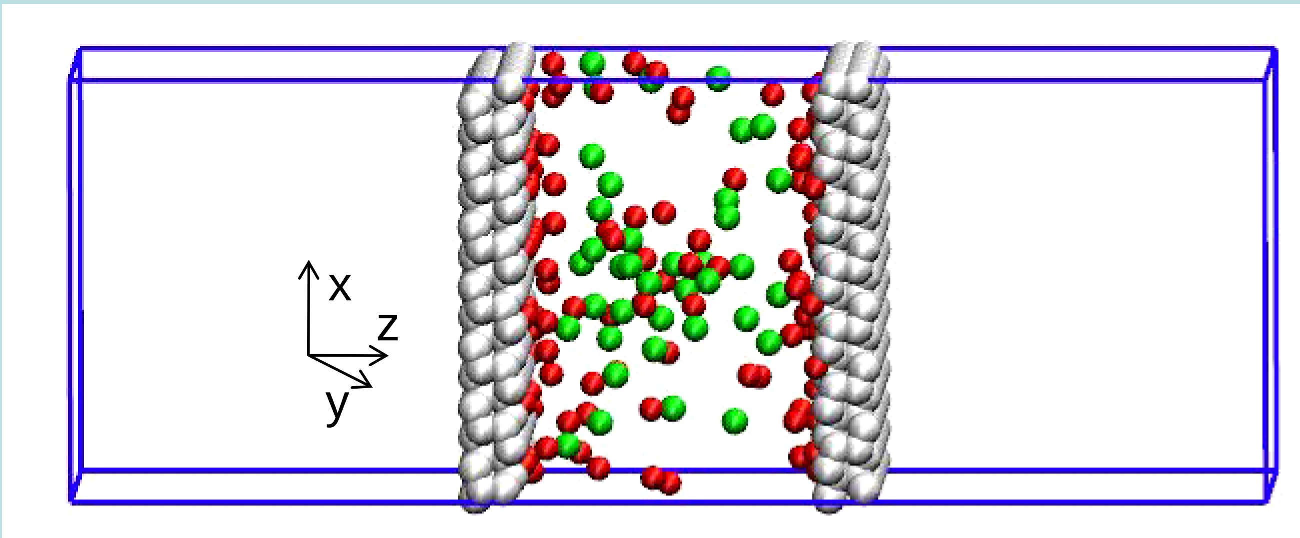


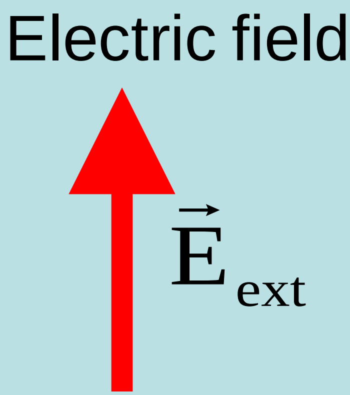
US Army Corps of Engineers

BUILDING STRONG®

MISSISSIPPI STATE UNIVERSITY™ CAVS

Electro-osmotic flow model





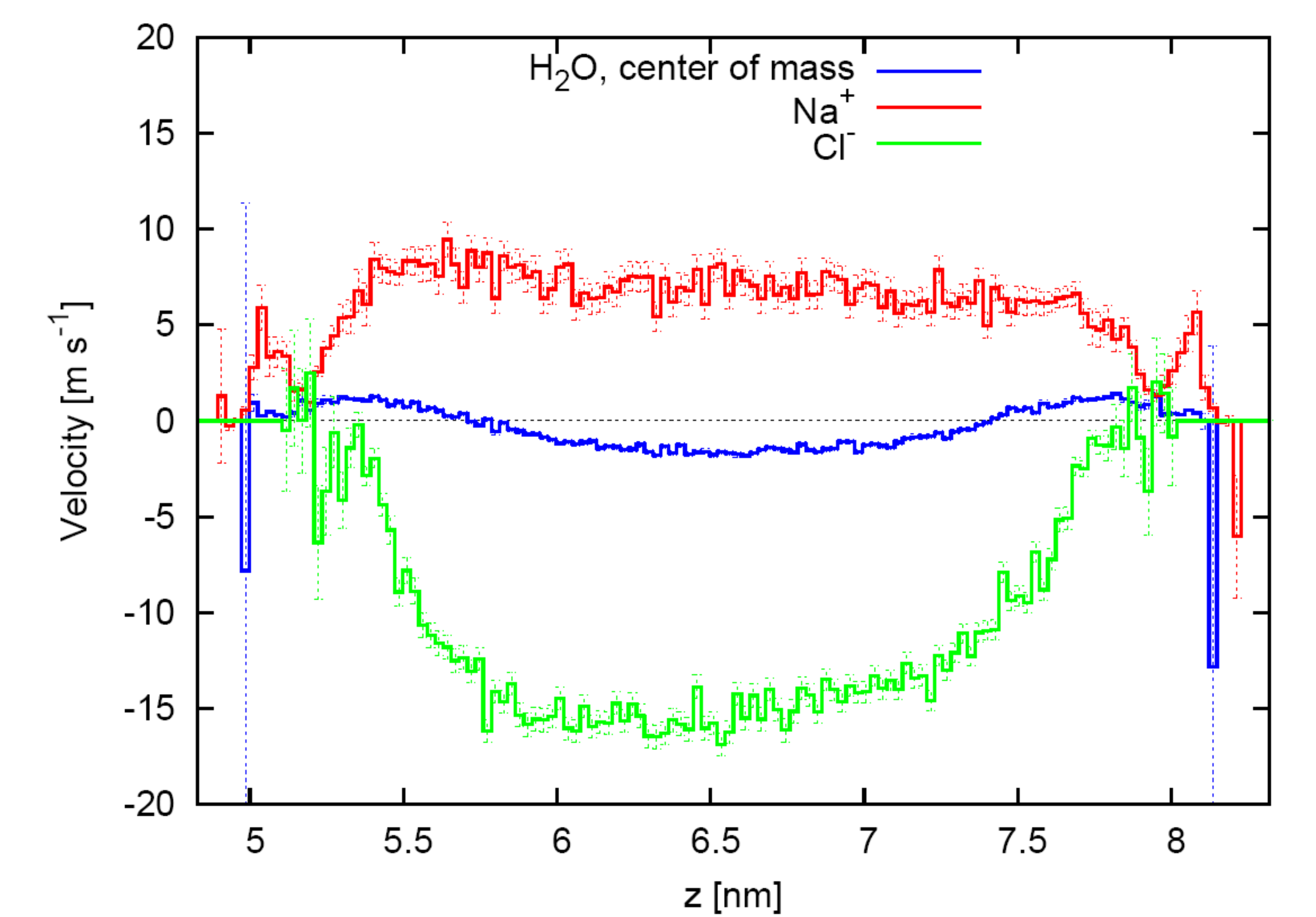
Fixed Si channel walls, innermost layer charged negatively

Dimensions of a solute region 4.66x4.22x3.49 nm, PBC x,y.

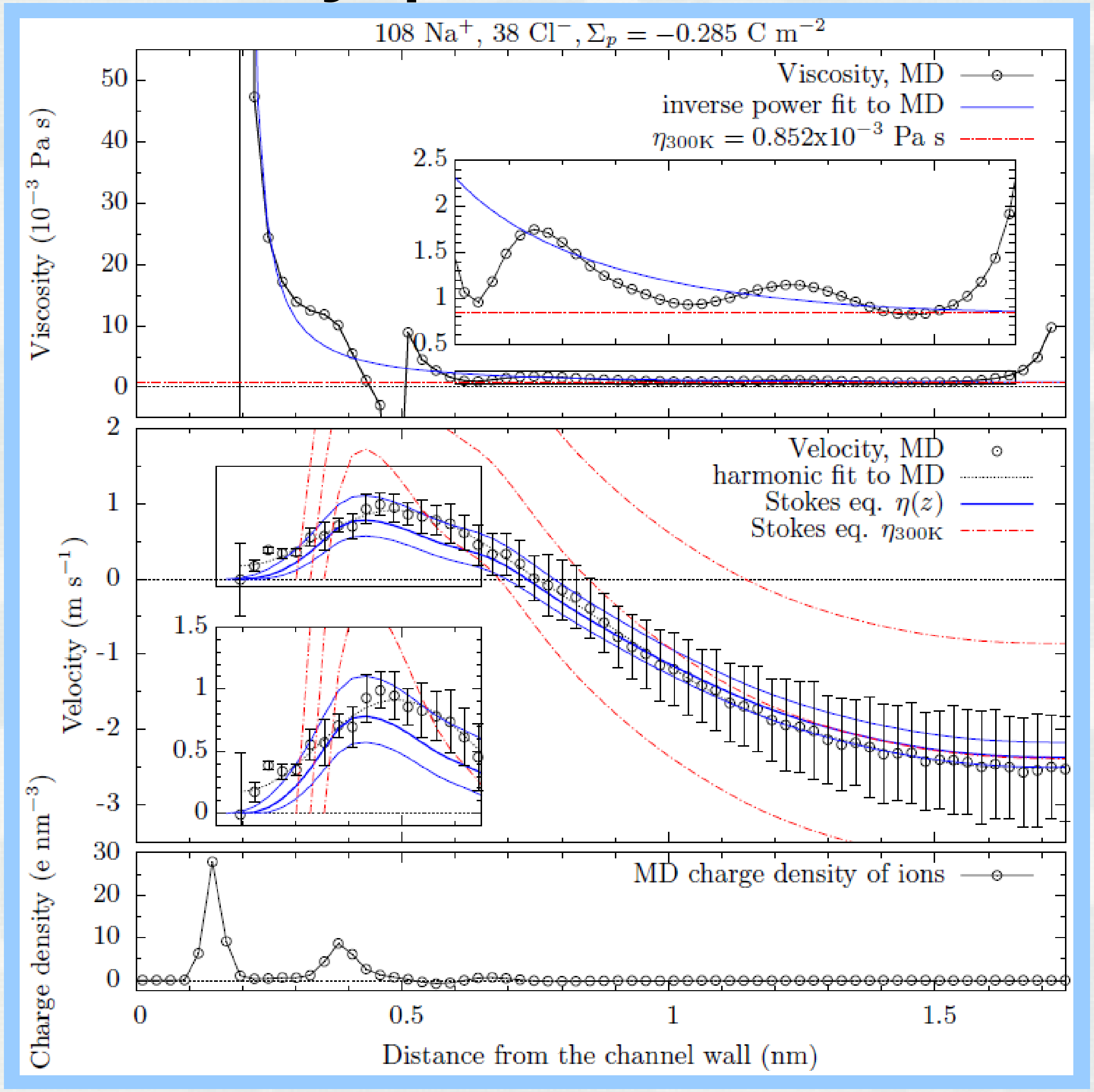
108 Na^+, 38 Cl^-, 2144 SPC/E H_2O molecules (not shown)

R. Qiao and N. R. Aluru: Charge Inversion and Flow Reversal in a Nanochannel Electro-osmotic Flow, PRL 92 (19) 2004

Velocity profiles



Velocity predicted from charge density



Stokes equation:

$$\frac{d}{dz}\left[\eta(z)\frac{du_x(z)}{dz}\right] = -F_d(z)$$

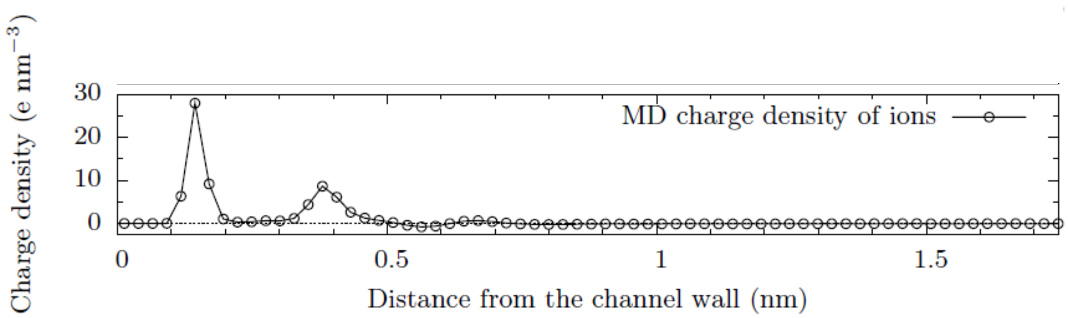
Blue:
inverse power viscosity

$$\eta(z) = \left[1 - \left(\frac{z}{h}\right)^2\right]^{-p} \eta_{\exp}$$

Red:
constant viscosity

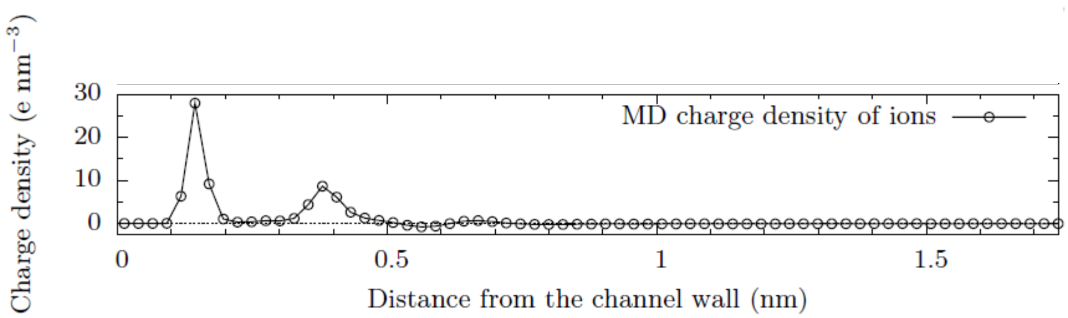
Black circles:
Molecular Dynamics

Velocity predicted from charge density



$$\mathbf{F}_d(z) = e\left[c_{\mathrm{Na}^+}(z) - c_{\mathrm{Cl}^-}(z)\right]\mathbf{E}_{ext}$$

Velocity predicted from charge density

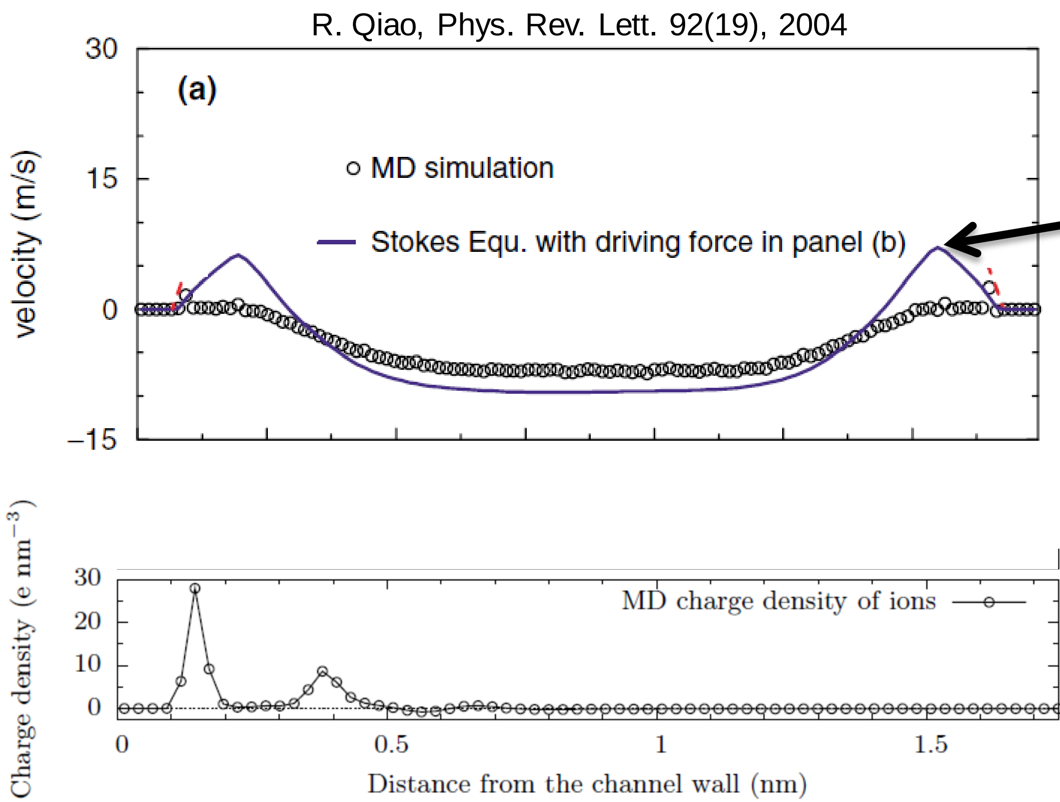


Stokes equation:

$$\frac{d}{dz}\left[\eta(z)\frac{du_x(z)}{dz}\right] = -F_d(z)$$

$$\mathbf{F}_d(z) = e\left[c_{\mathrm{Na}^+}(z) - c_{\mathrm{Cl}^-}(z)\right]\mathbf{E}_{ext}$$

Velocity predicted from charge density

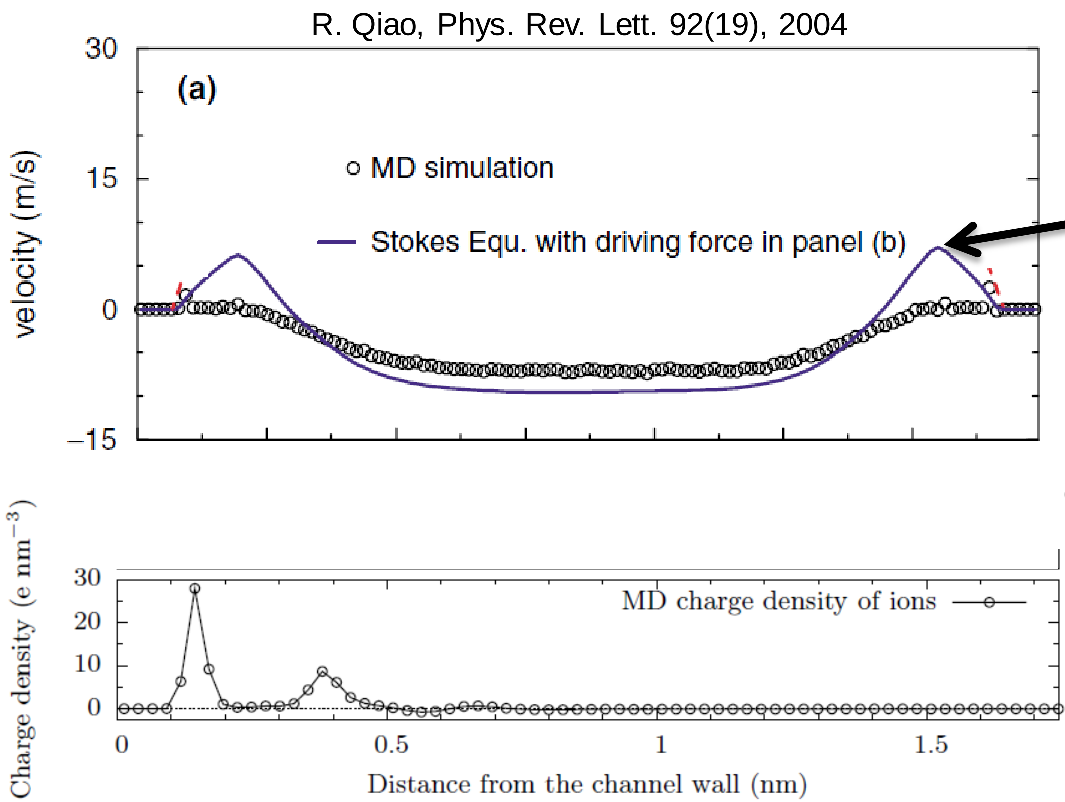


Stokes equation:

$$\frac{d}{dz}\left[\eta(z)\frac{du_x(z)}{dz}\right] = -F_d(z)$$

Dark blue line:
velocity prediction
from MD charge density

Velocity predicted from charge density

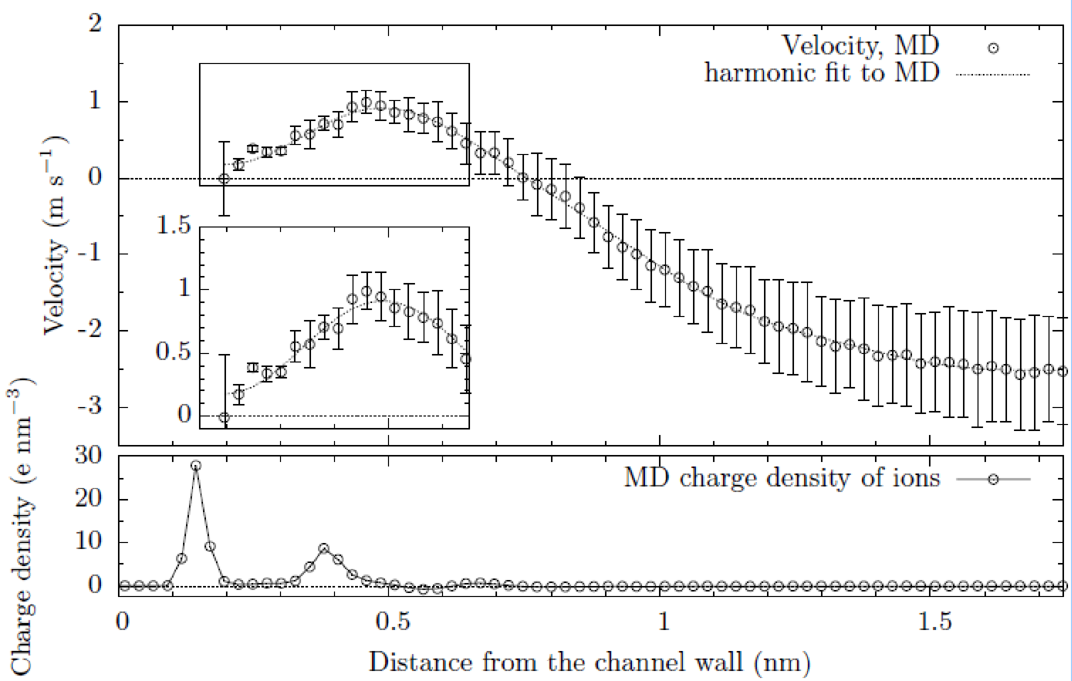


Stokes equation:

$$\frac{d}{dz}\left[\eta(z)\frac{du_x(z)}{dz}\right] = -F_d(z)$$

Dark blue line:
velocity prediction
from MD charge density,
assumes constant viscosity

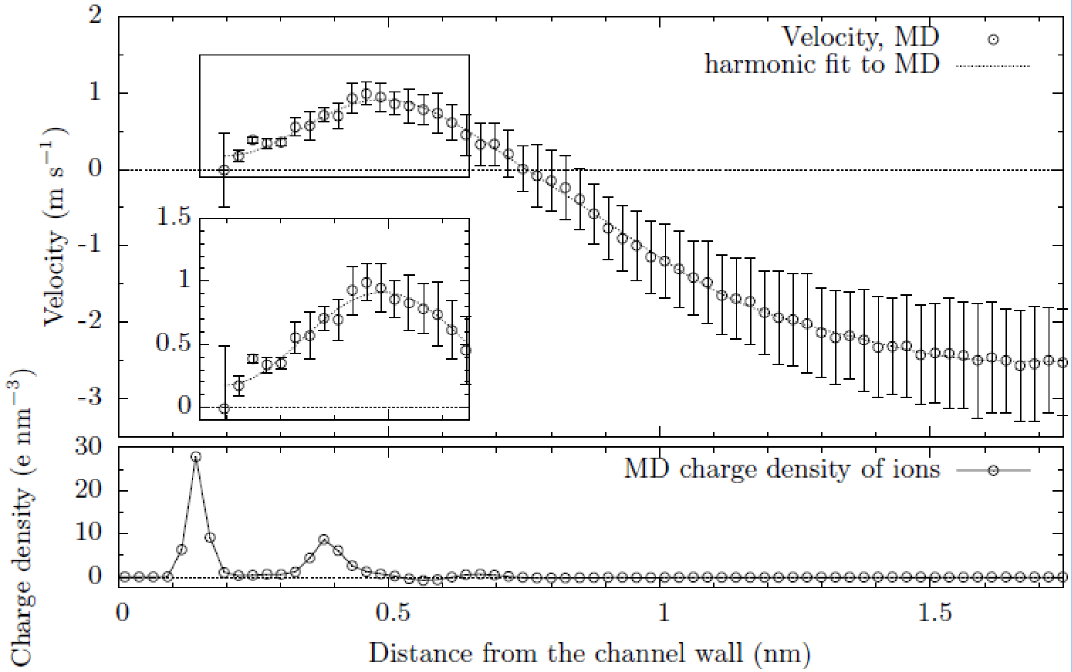
Viscosity estimation



Stokes equation:

$$\frac{d}{dz}\left[\eta(z)\frac{du_x(z)}{dz}\right] = -F_d(z)$$

Viscosity estimation



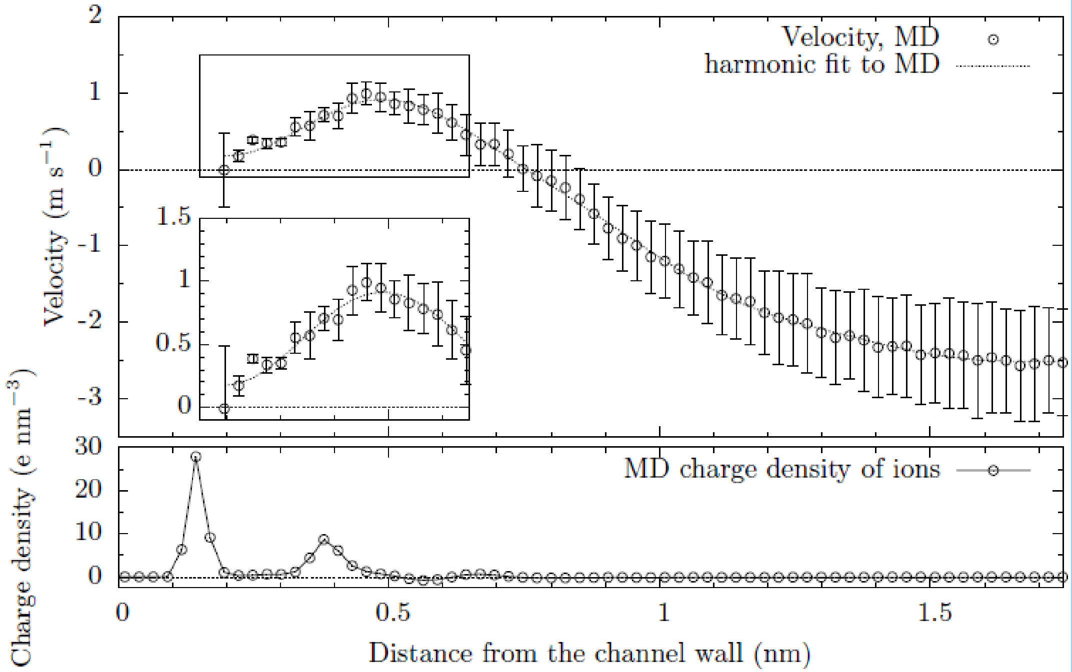
Stokes equation:

$$\frac{d}{dz}\left[\eta(z)\frac{du_x(z)}{dz}\right] = -F_d(z)$$

Integrated:

$$\eta(z)|_{z=z_0} = \frac{-\int_0^{z_0} F_d(z)\,dz}{\left.\frac{du_x(z)}{dz}\right|_{z=z_0}}$$

Viscosity estimation



Stokes equation:

$$\frac{d}{dz}\left[\eta(z)\frac{du_x(z)}{dz}\right] = -F_d(z)$$

Integrated:

$$\eta(z)|_{z=z_0} = \frac{-\int_0^{z_0} F_d(z)\, dz}{\left.\frac{du_x(z)}{dz}\right|_{z=z_0}}$$

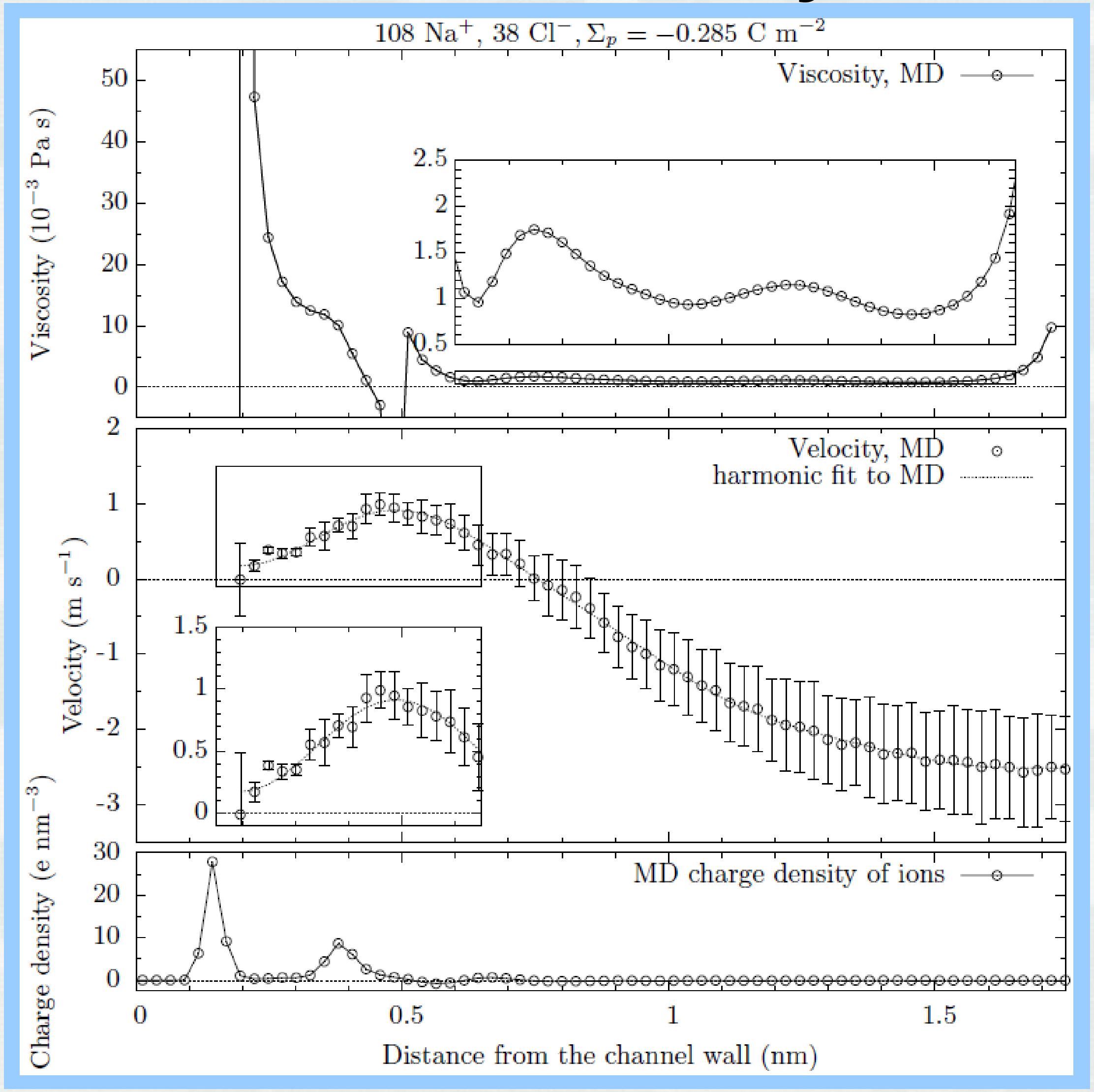
Velocity approximation:

$$u_{xfit}(z) = \sum_{n=0}^{7} a_n \cos\left(n\pi\frac{z}{h}\right)$$

J.B. Freund, J. Chem. Phys. 116(5), 2002

$$u_{\text{fit}}(y) = u_m \exp\left[\frac{(y-y_m)^4}{y_1^4}\right] + \sum_{n=0}^{11} a_n \cos\frac{\pi y n}{L}$$

Viscosity estimation



Stokes equation:

$$\frac{d}{dz}\left[\eta(z)\frac{du_x(z)}{dz}\right] = -F_d(z)$$

Integrated:

$$\eta(z)|_{z=z_0} = \frac{-\int_0^{z_0} F_d(z)\,dz}{\left.\frac{du_x(z)}{dz}\right|_{z=z_0}}$$

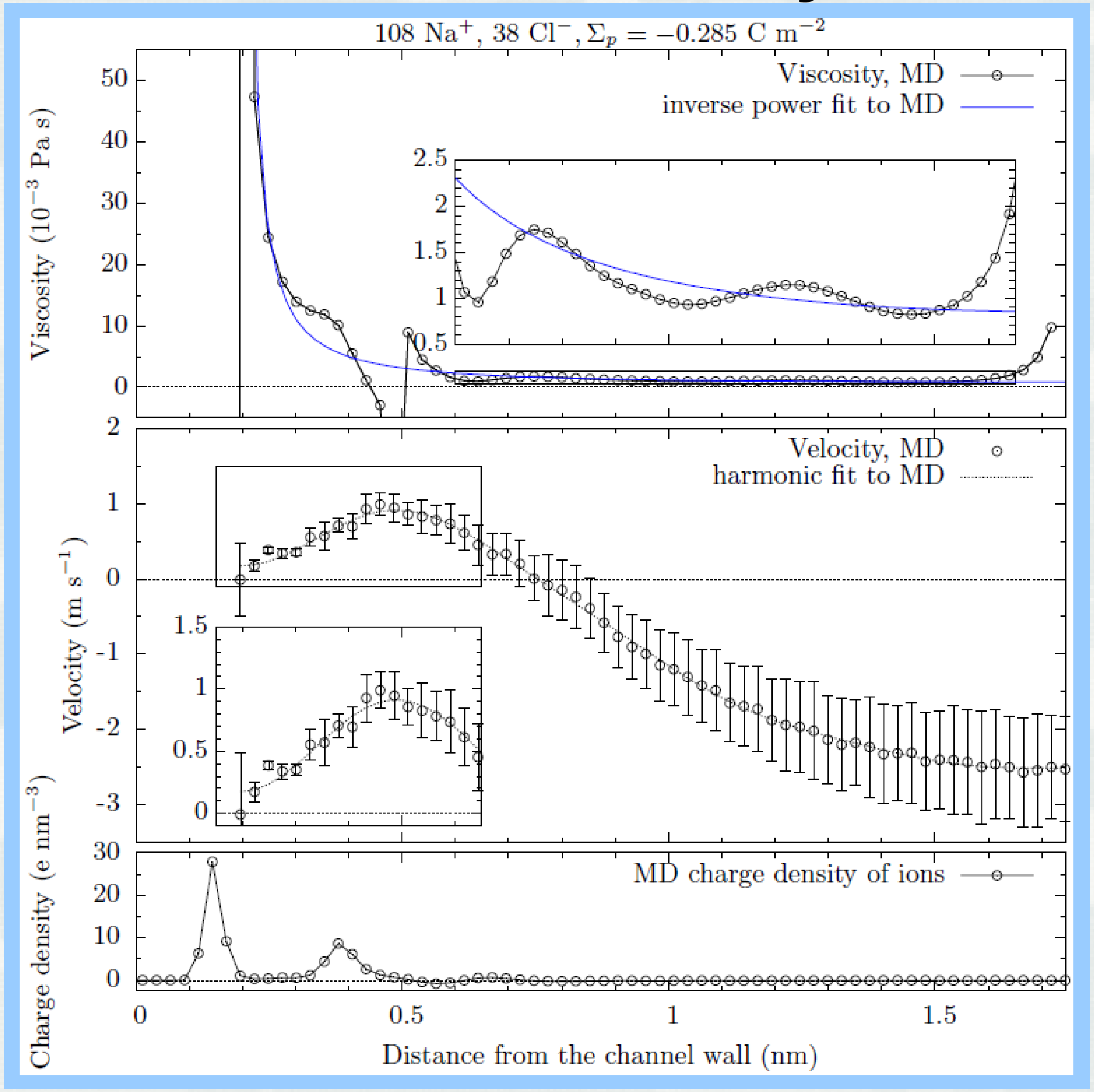
Velocity approximation:

$$u_{x\,fit}(z) = \sum_{n=0}^{7} a_n \cos\left(n\pi\frac{z}{h}\right)$$

J.B. Freund, J. Chem. Phys. 116(5), 2002

$$u_{\mathrm{fit}}(y) = u_m \exp\left[\frac{(y-y_m)^4}{y_1^4}\right] + \sum_{n=0}^{11} a_n \cos\frac{\pi y n}{L}$$

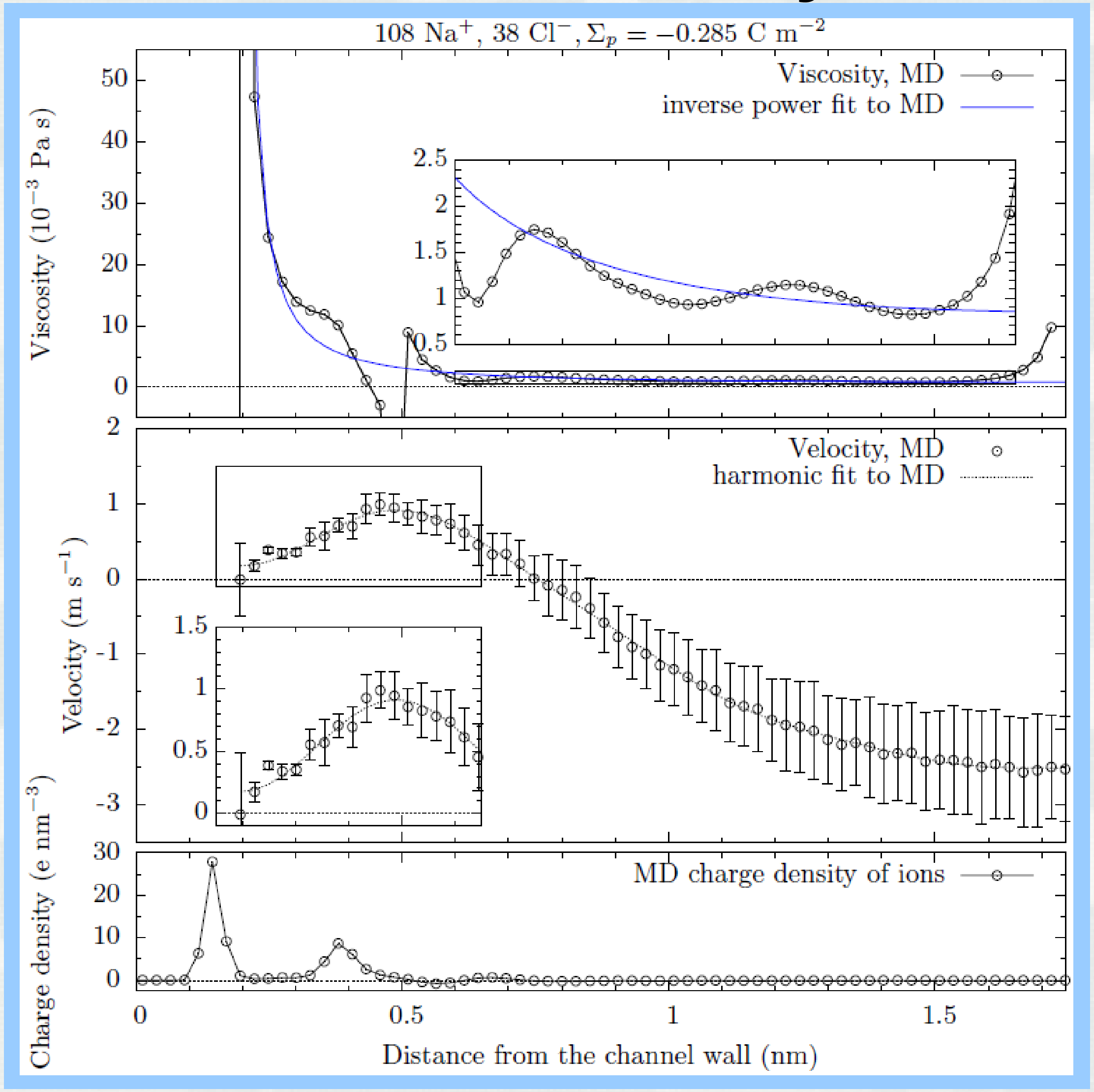
Viscosity estimation



Stokes equation:

$$\frac{d}{dz}\left[\eta(z)\frac{du_x(z)}{dz}\right] = -F_d(z)$$

Viscosity estimation



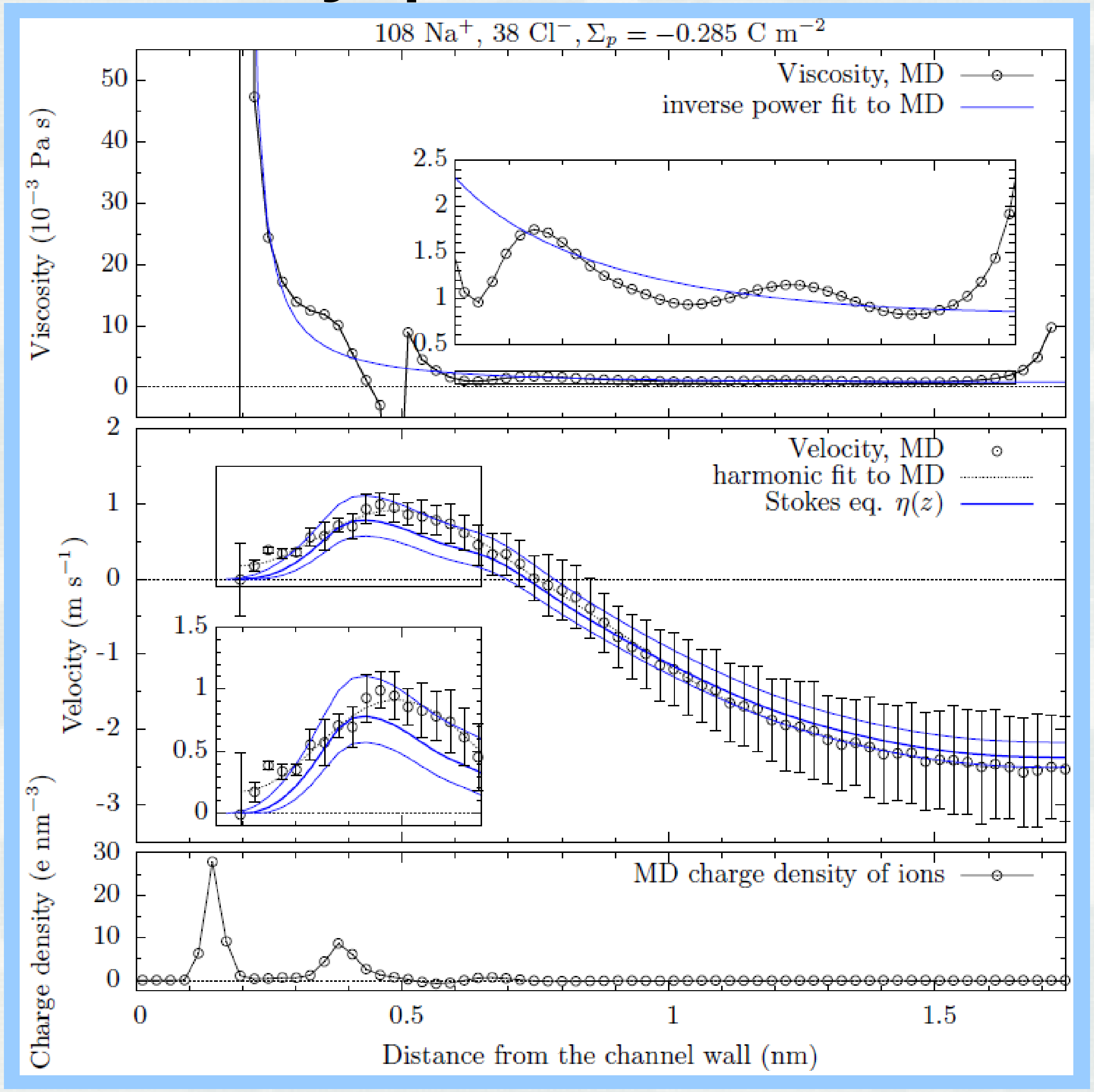
Stokes equation:

$$\frac{d}{dz}\left[\eta(z)\frac{du_x(z)}{dz}\right] = -F_d(z)$$

Blue:
inverse power viscosity

$$\eta(z) = \left[1 - \left(\frac{z}{h}\right)^2\right]^{-p} \eta_{\exp}$$

Velocity predicted from charge density



Stokes equation:

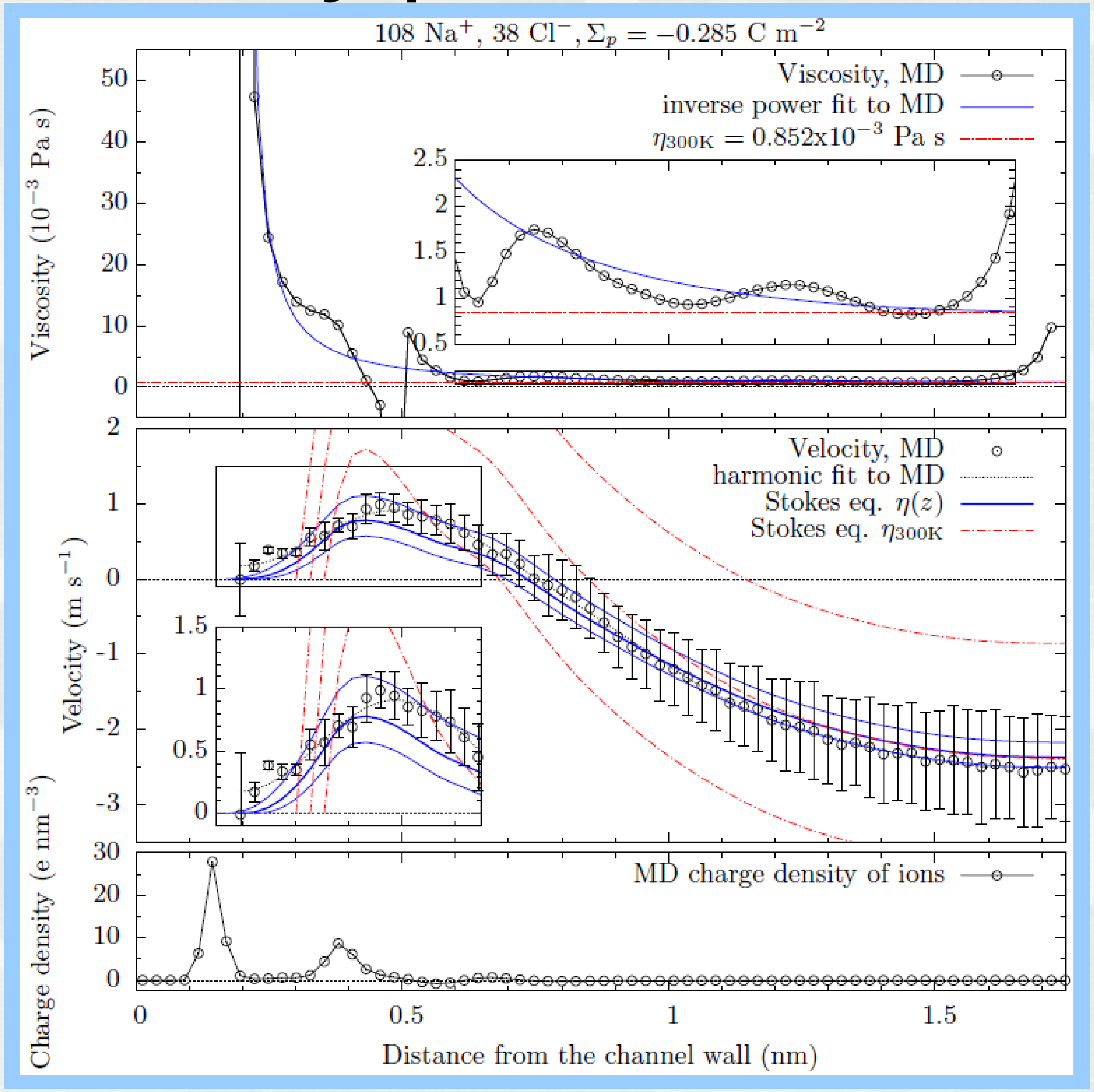
$$\frac{d}{dz}\left[\eta(z)\frac{du_x(z)}{dz}\right] = -F_d(z)$$

Blue:

inverse power viscosity

$$\eta(z) = \left[1 - \left(\frac{z}{h}\right)^2\right]^{-p} \eta_{\mathrm{exp}}$$

Velocity predicted from charge density



Stokes equation:

$$\frac{d}{dz}\left[\eta(z)\frac{du_x(z)}{dz}\right] = -F_d(z)$$

Blue:

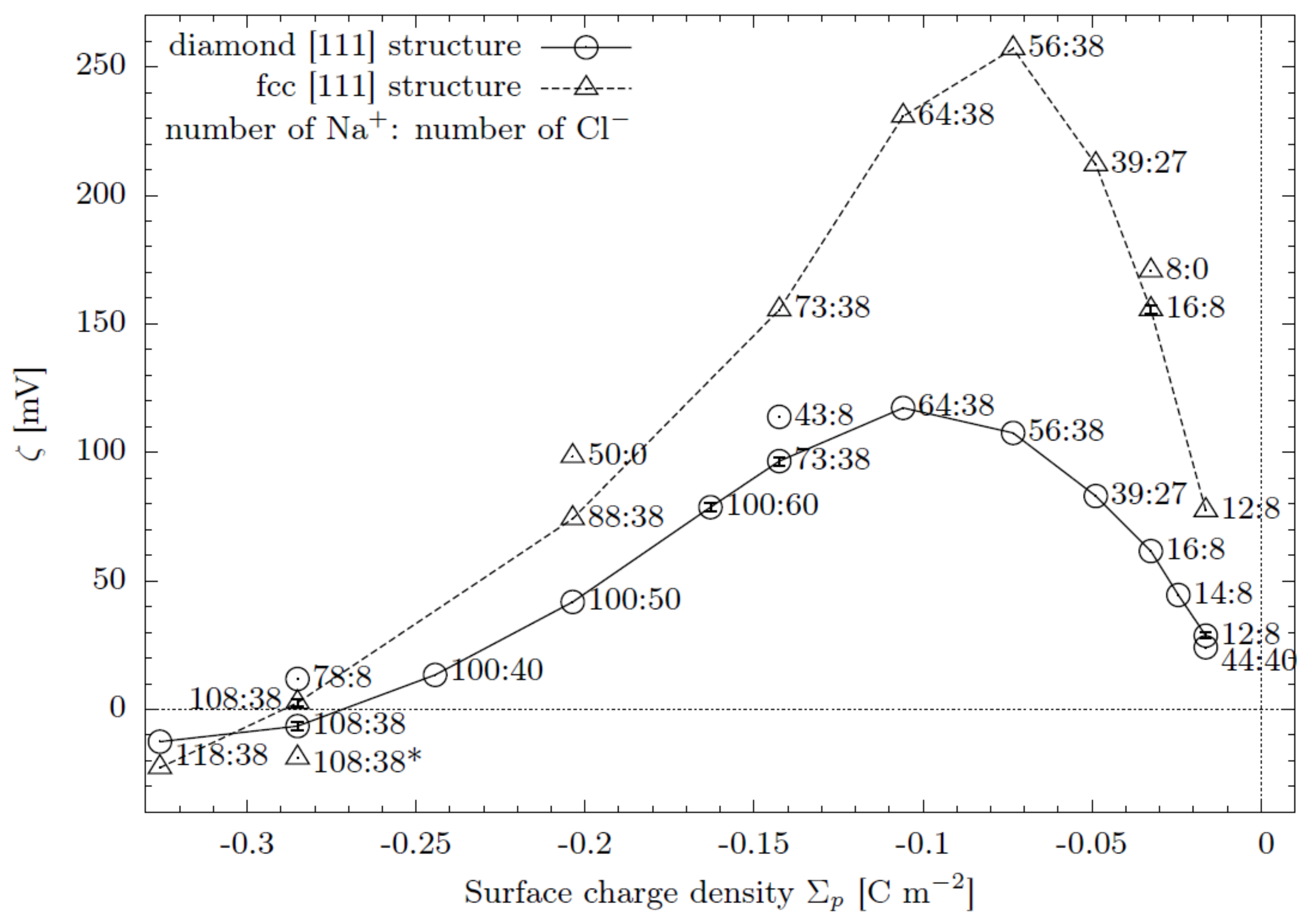
inverse power viscosity

$$\eta(z) = \left[1 - \left(\frac{z}{h}\right)^2\right]^{-p} \eta_{\mathrm{exp}}$$

Red:

constant viscosity

Zeta potentials vs. surf. charge density for uniform partial surface charge



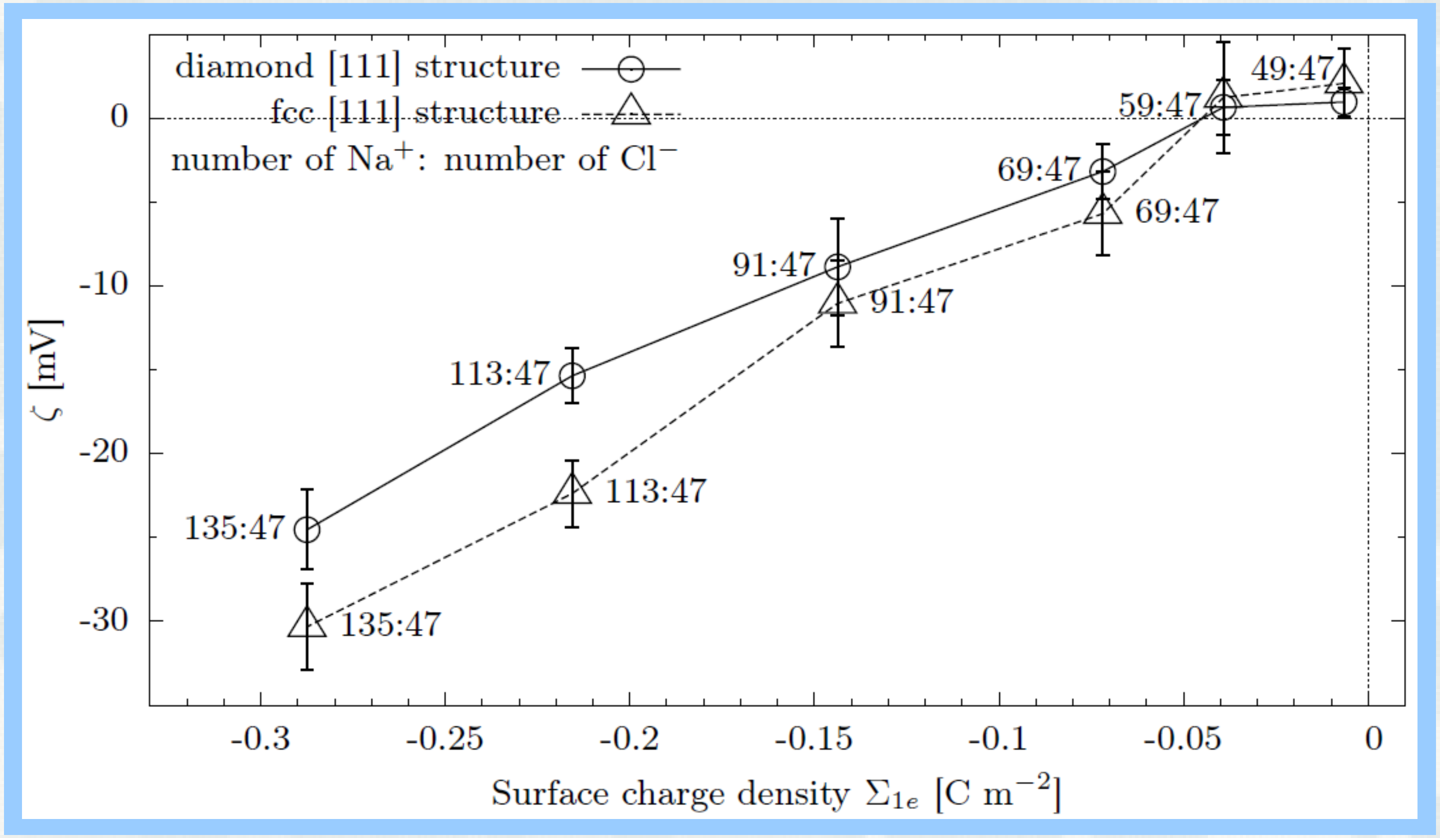
MD Zeta potential:

$$\zeta = \frac{u_x(z_{\text{center}})\eta}{\varepsilon_0 \varepsilon_r \mathrm{E}_x}$$

Zeta potential is proportional to the water velocity in the channel center.

Assumes u_x is linear in E_x.

Zeta potentials vs. surf. charge density for discrete partial surface charge



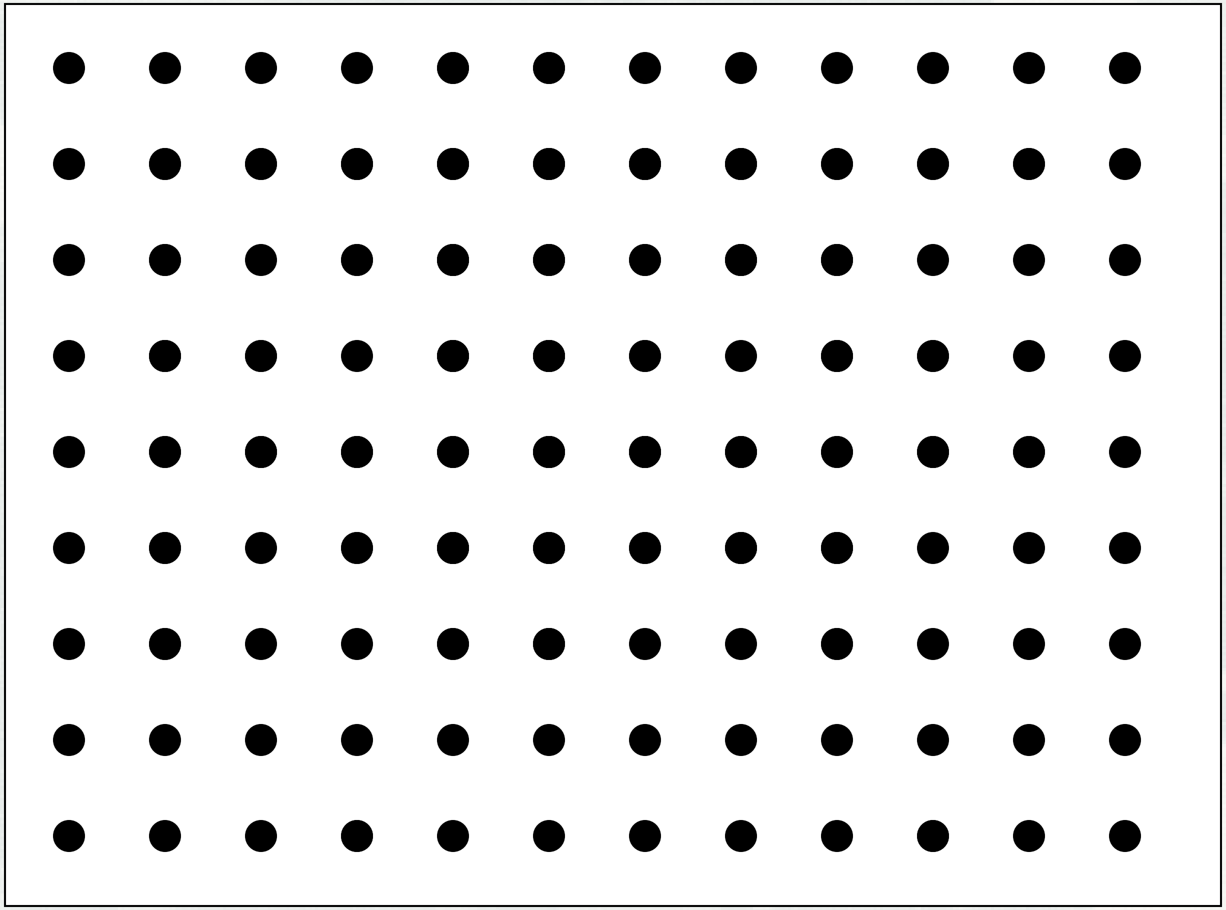
MD Zeta potential:

$$\zeta = \frac{u_x(z_{\text{center}})\eta}{\varepsilon_0 \varepsilon_r E_x}$$

Zeta potential is proportional to the water velocity in the channel center.

Assumes u_x is linear in E_x.

LBM parallelization



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

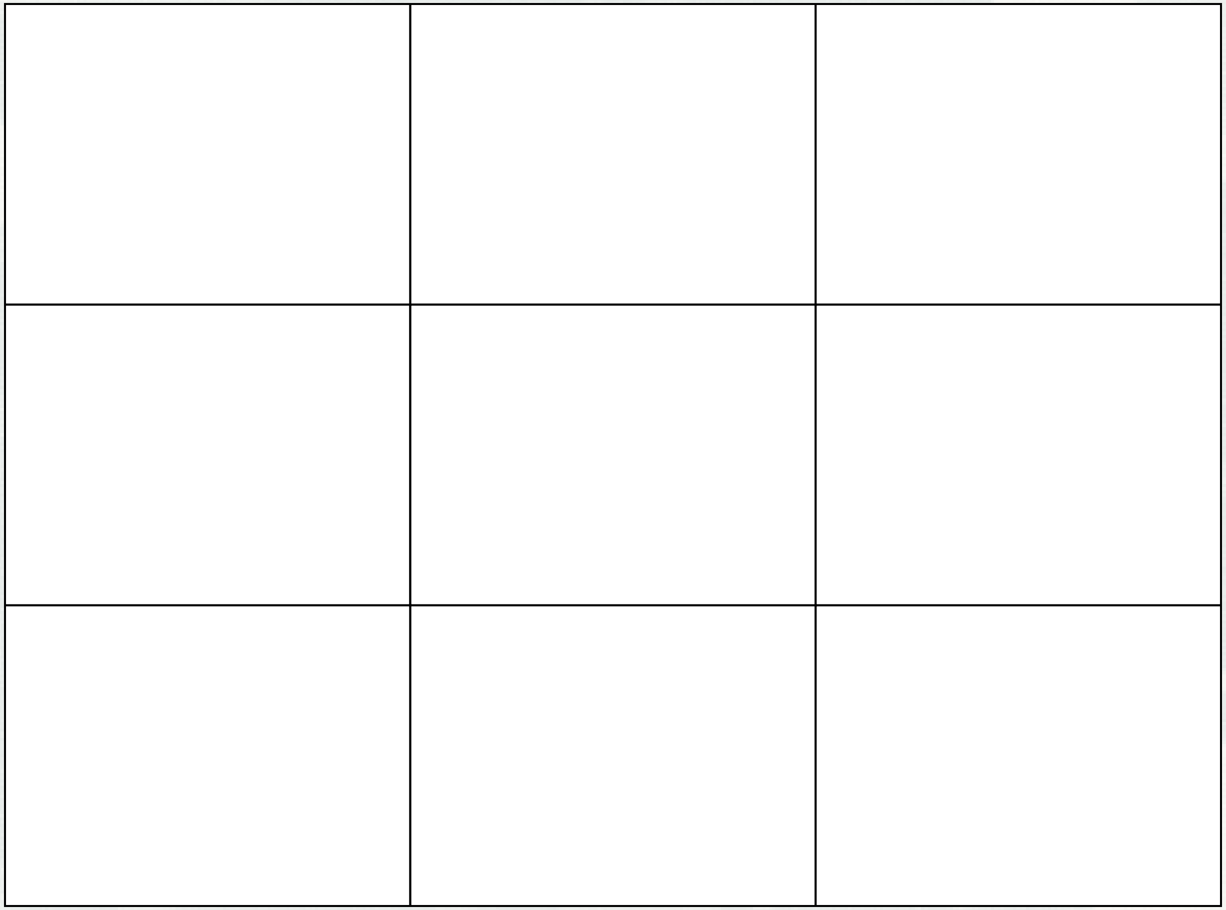
Acknowledgment: Mohsen Eshraghi provided excellent code guide and paralleliztion suggestions.

LBM parallelization

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

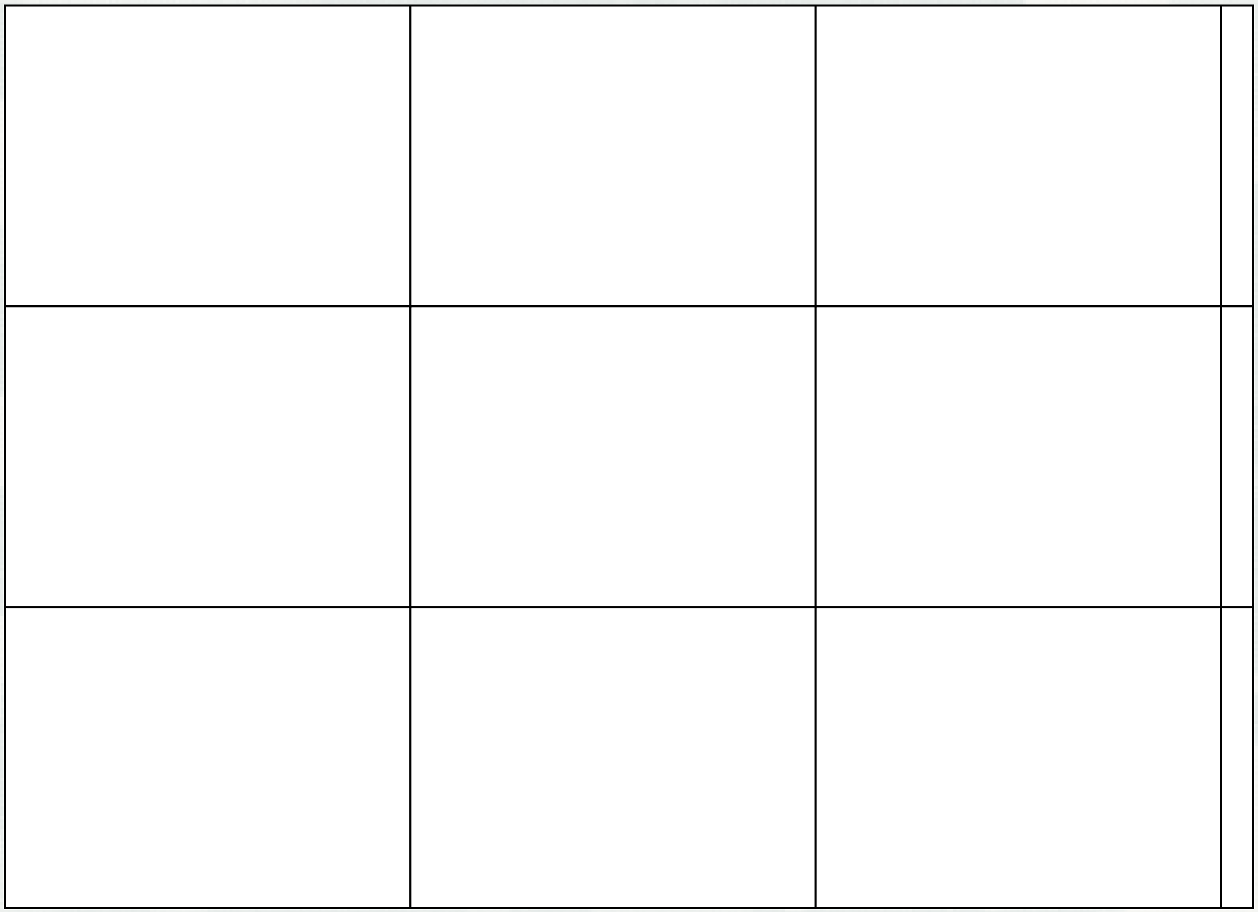
Spatial domain decomposition

LBM parallelization – streaming



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

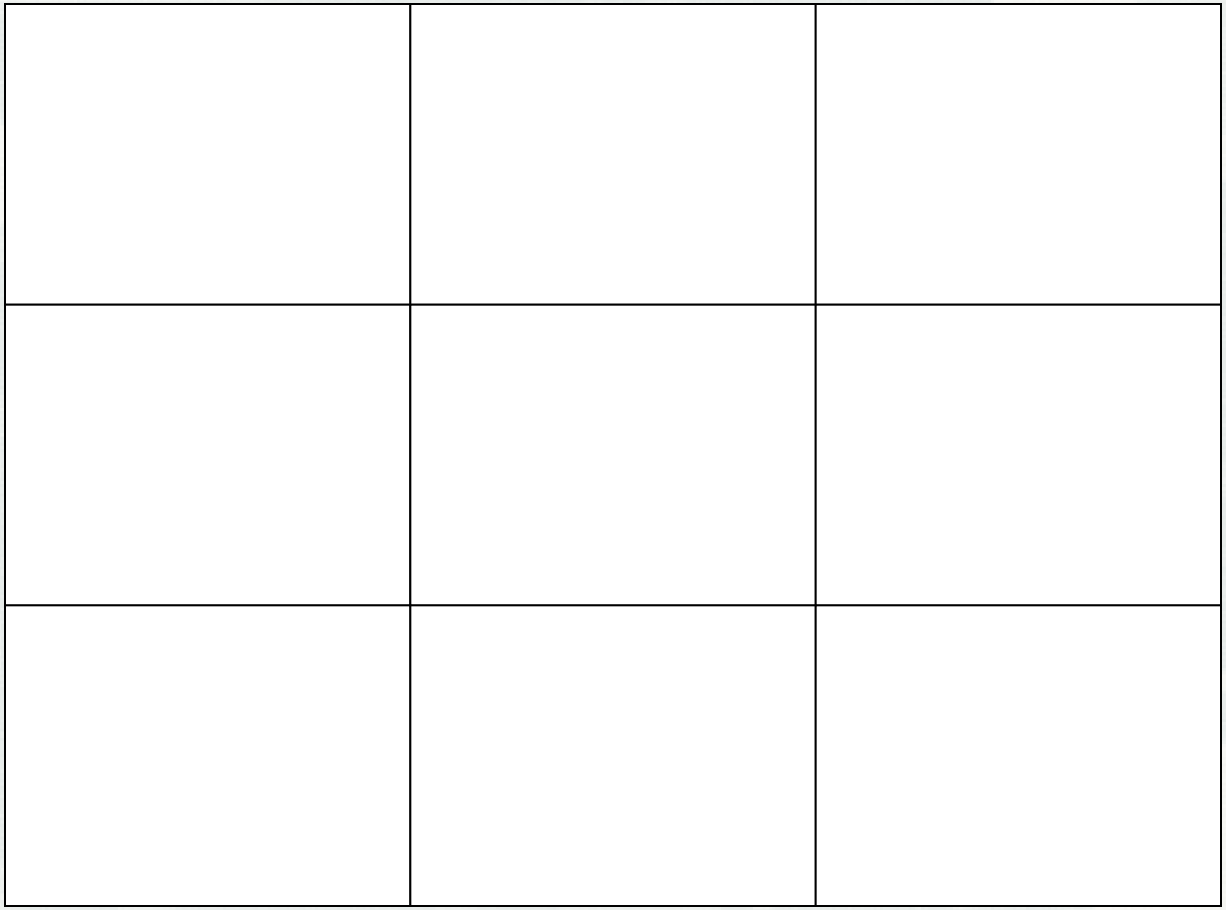
LBM parallelization – streaming



Direction

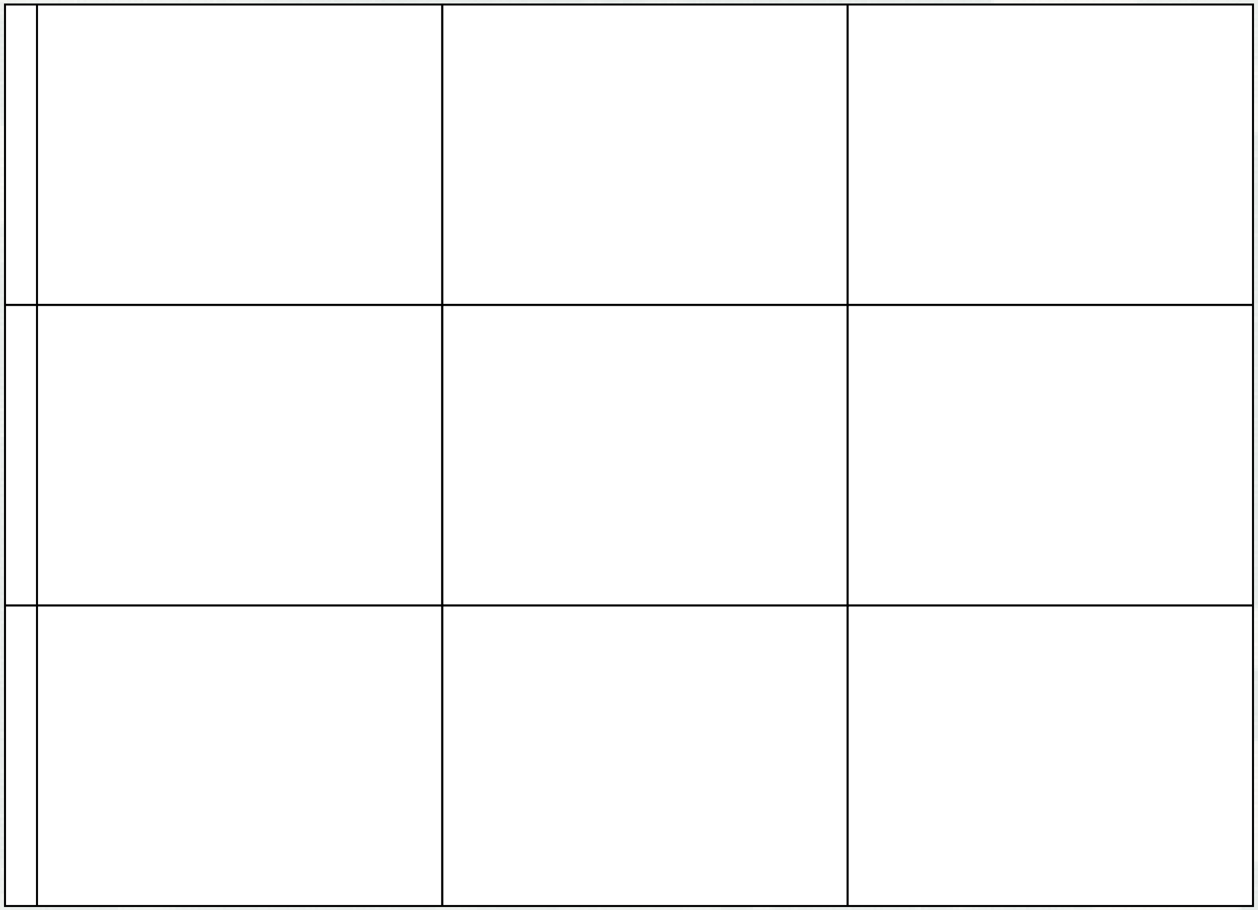
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)

LBM parallelization – streaming



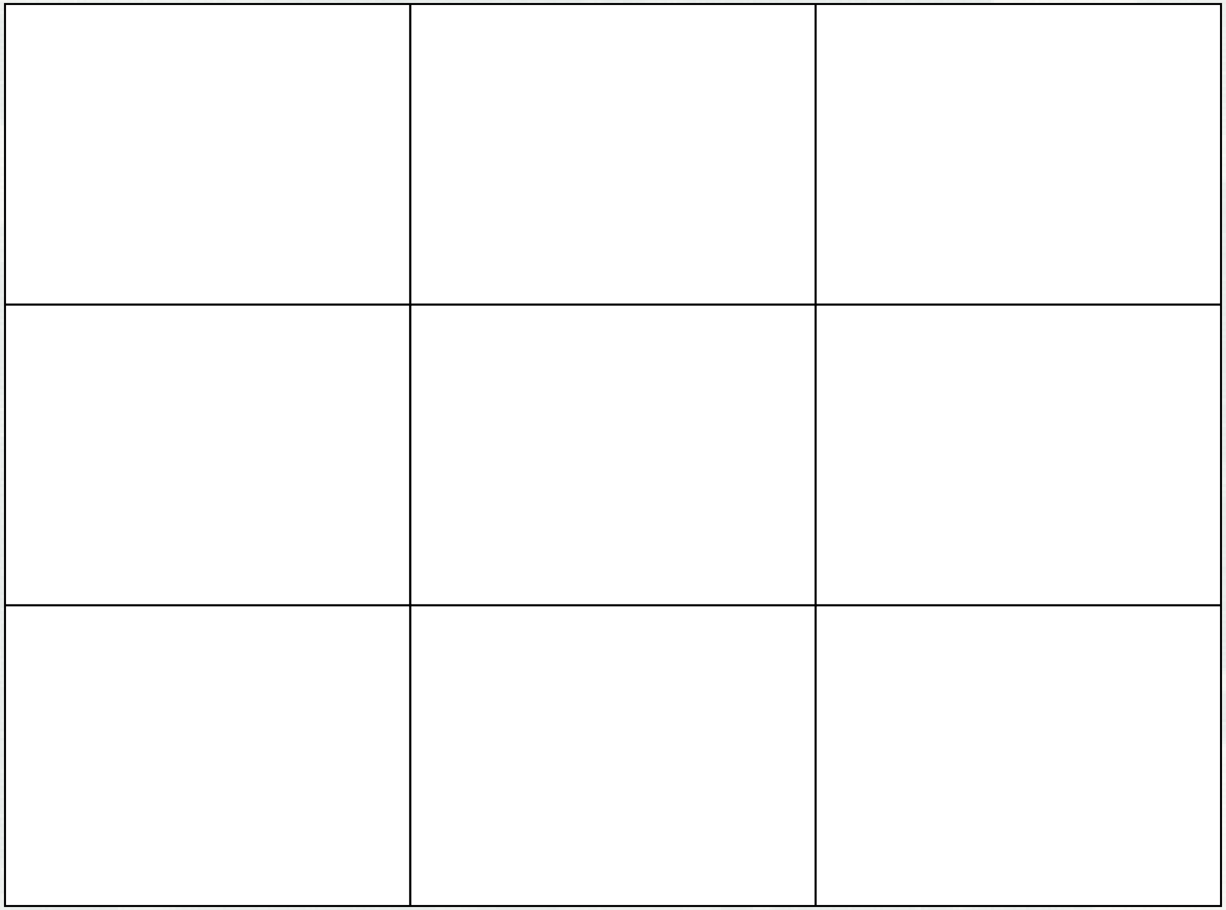
Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)

LBM parallelization – streaming



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

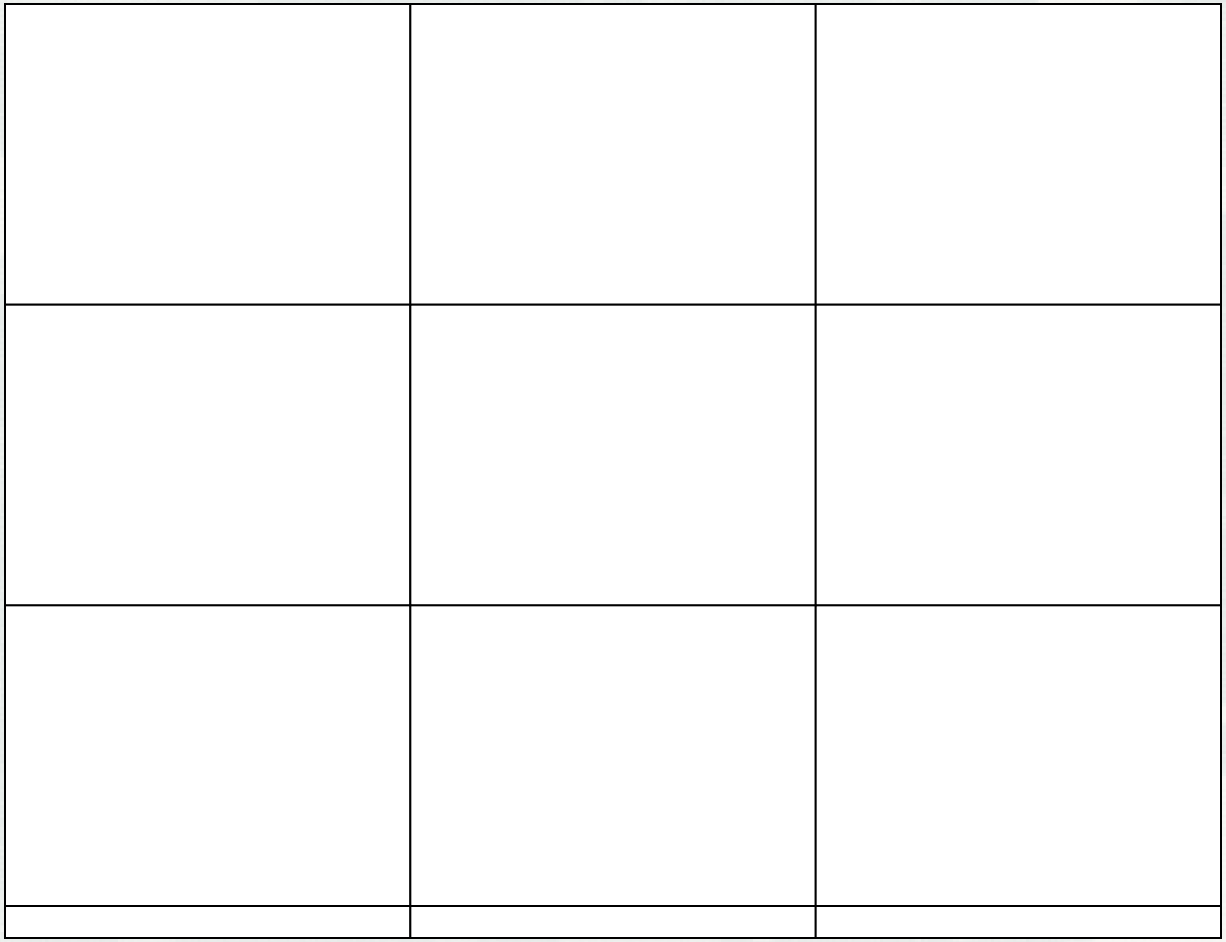
LBM parallelization – streaming



Direction

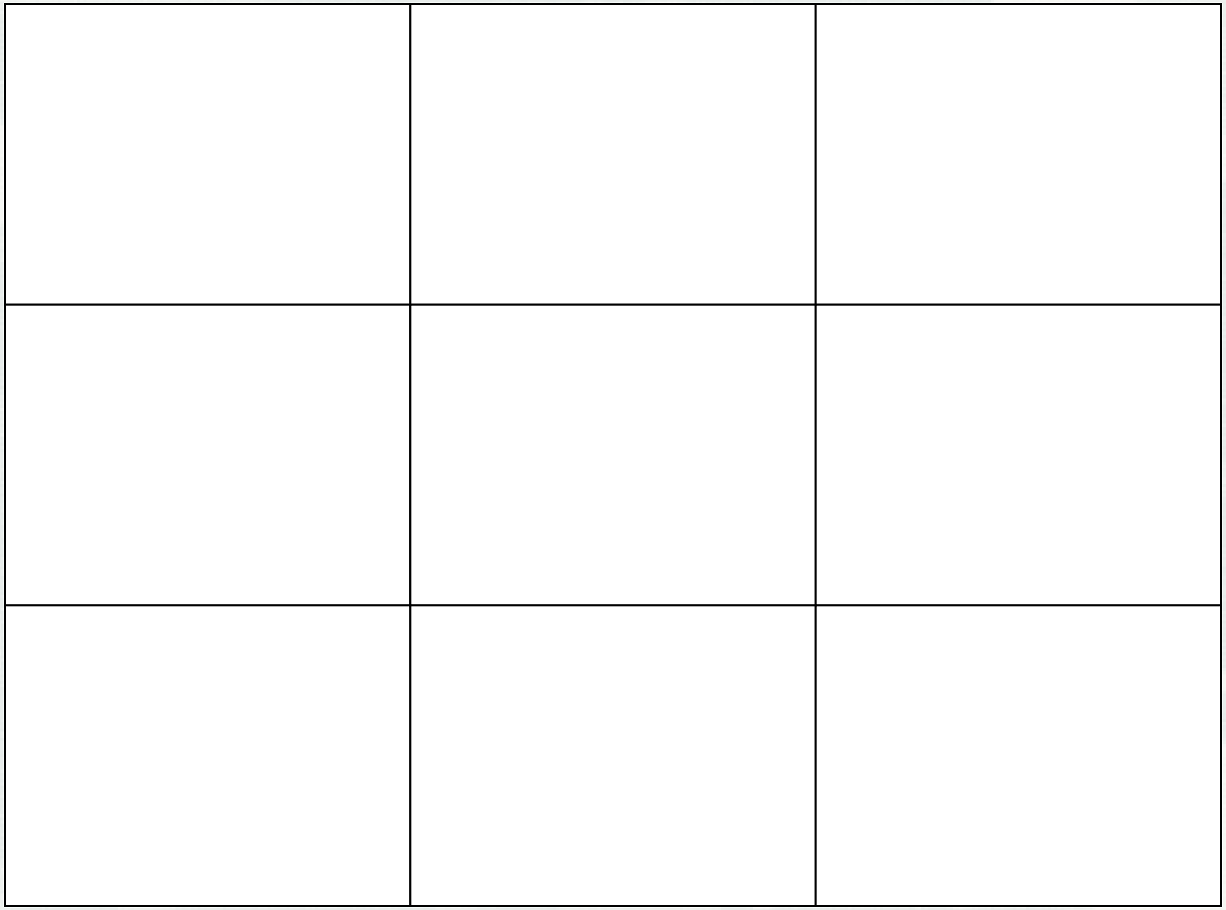
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)

LBM parallelization – streaming



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

LBM parallelization – streaming



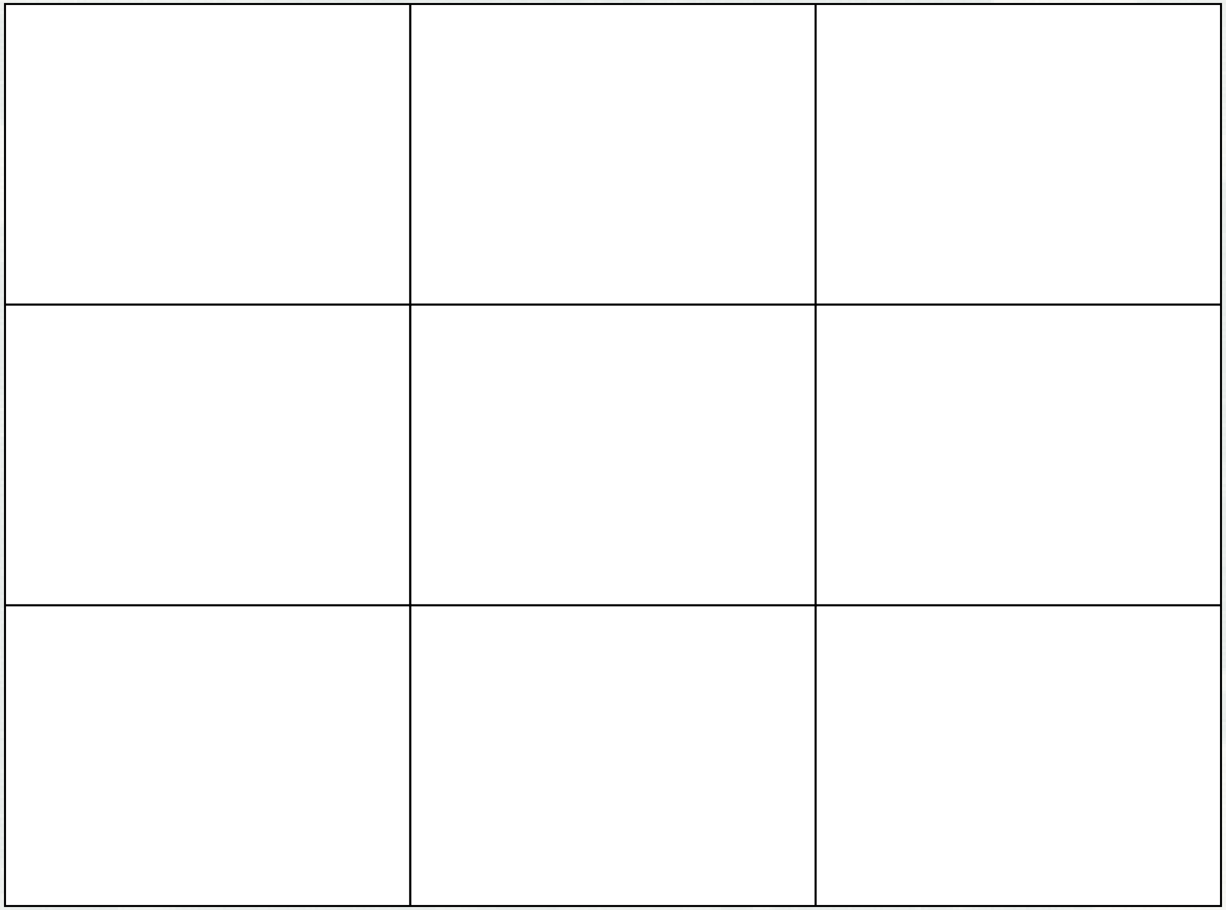
Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)

LBM parallelization – streaming

Direction

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

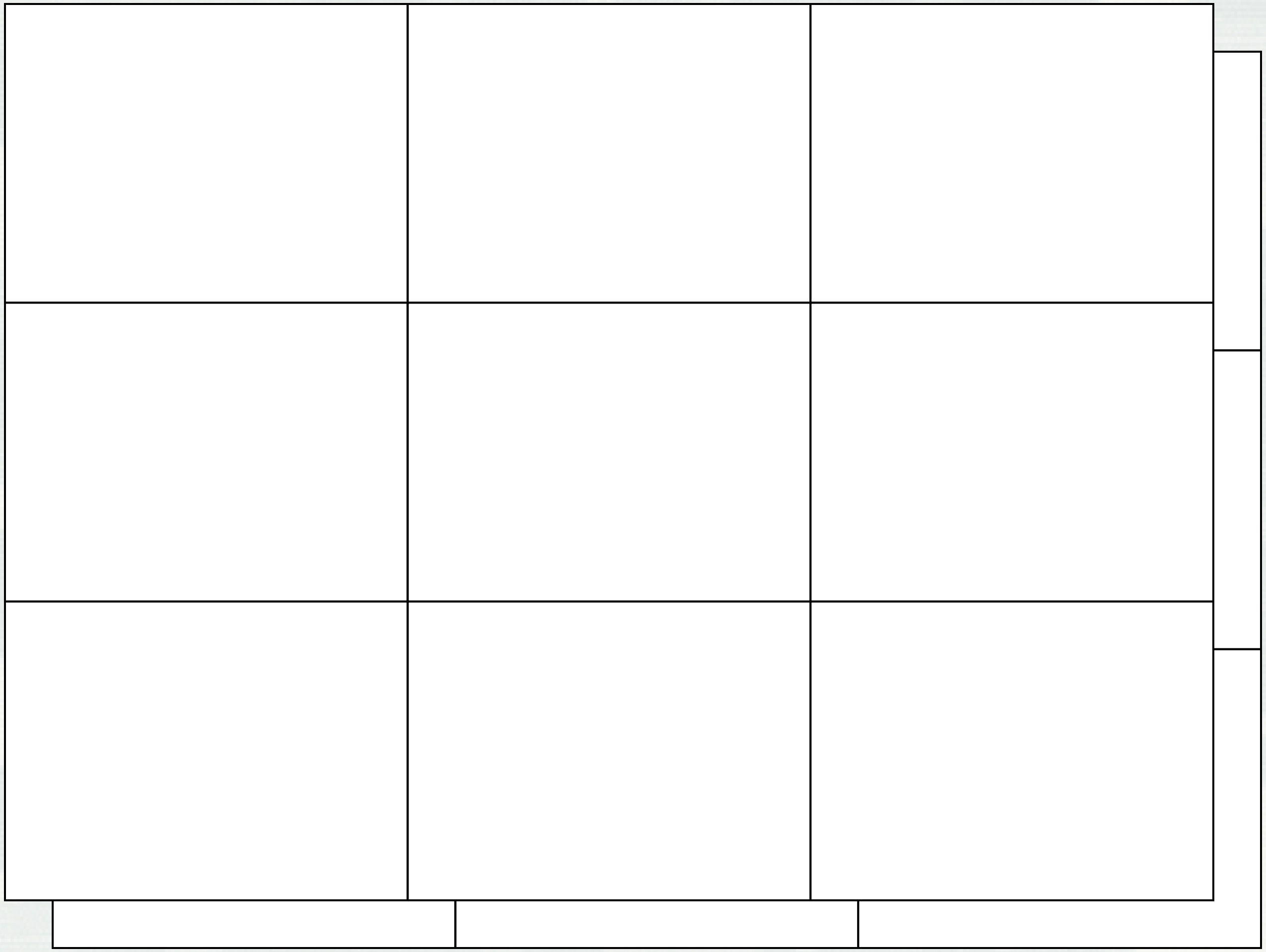
LBM parallelization – streaming



Direction

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

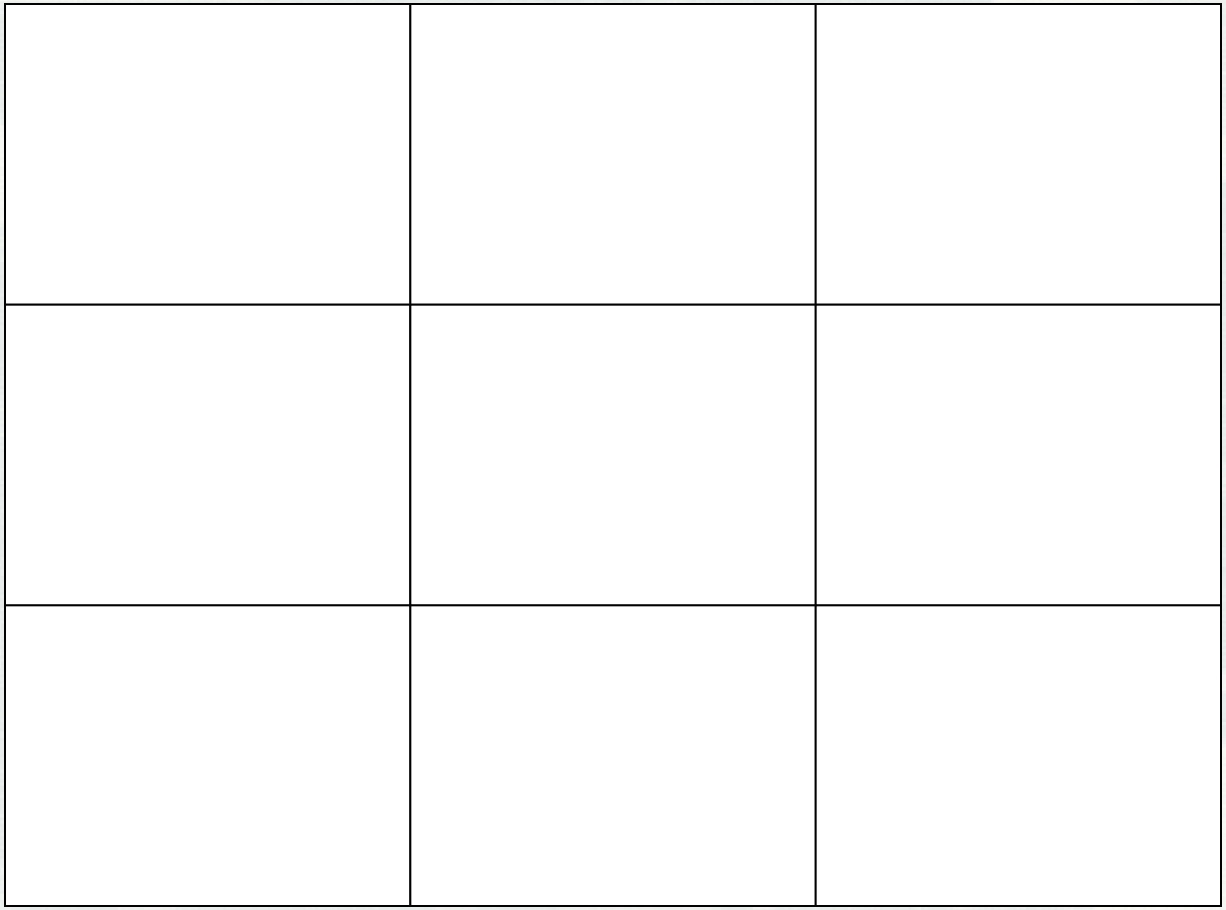
LBM parallelization – streaming



Direction

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

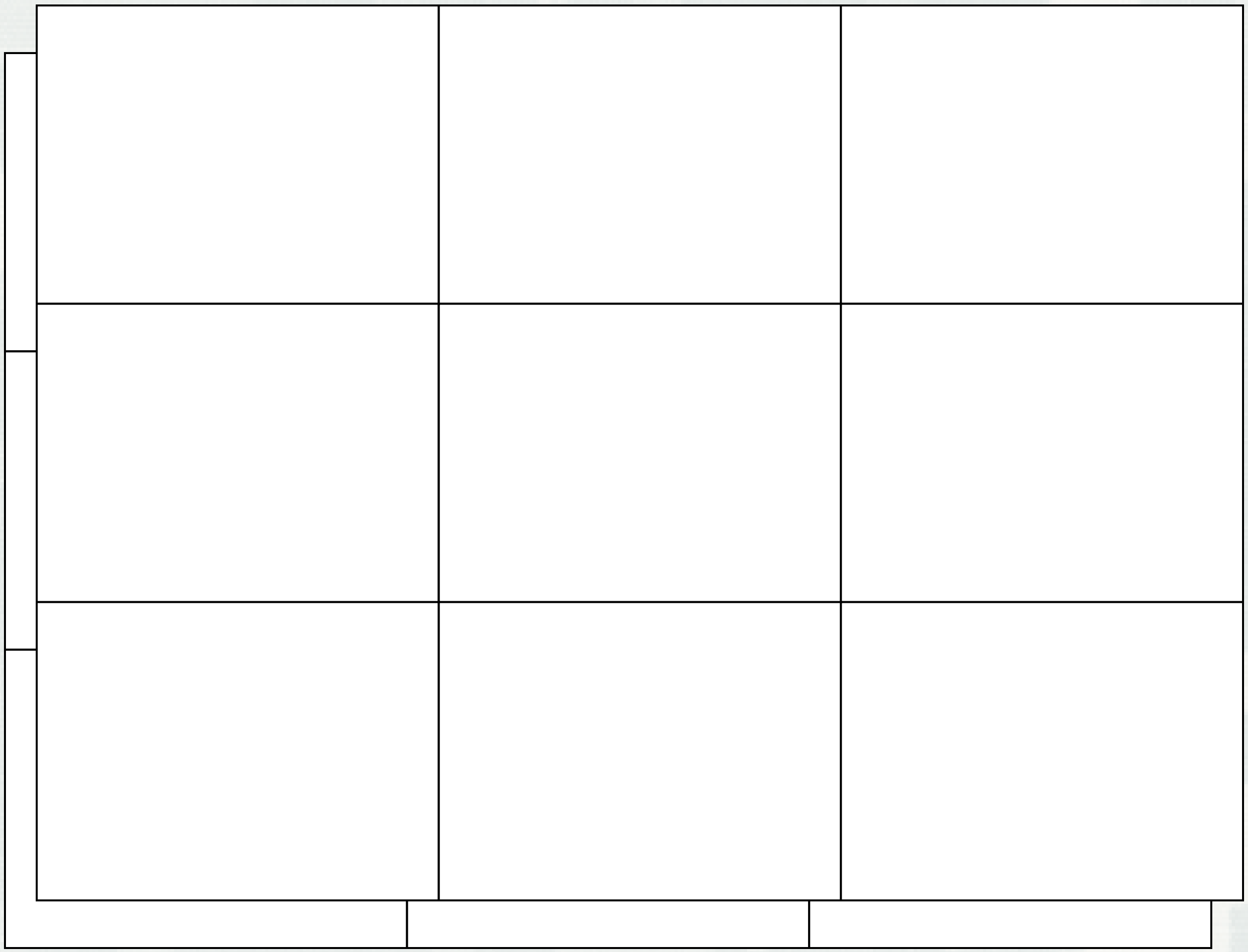
LBM parallelization – streaming



Direction

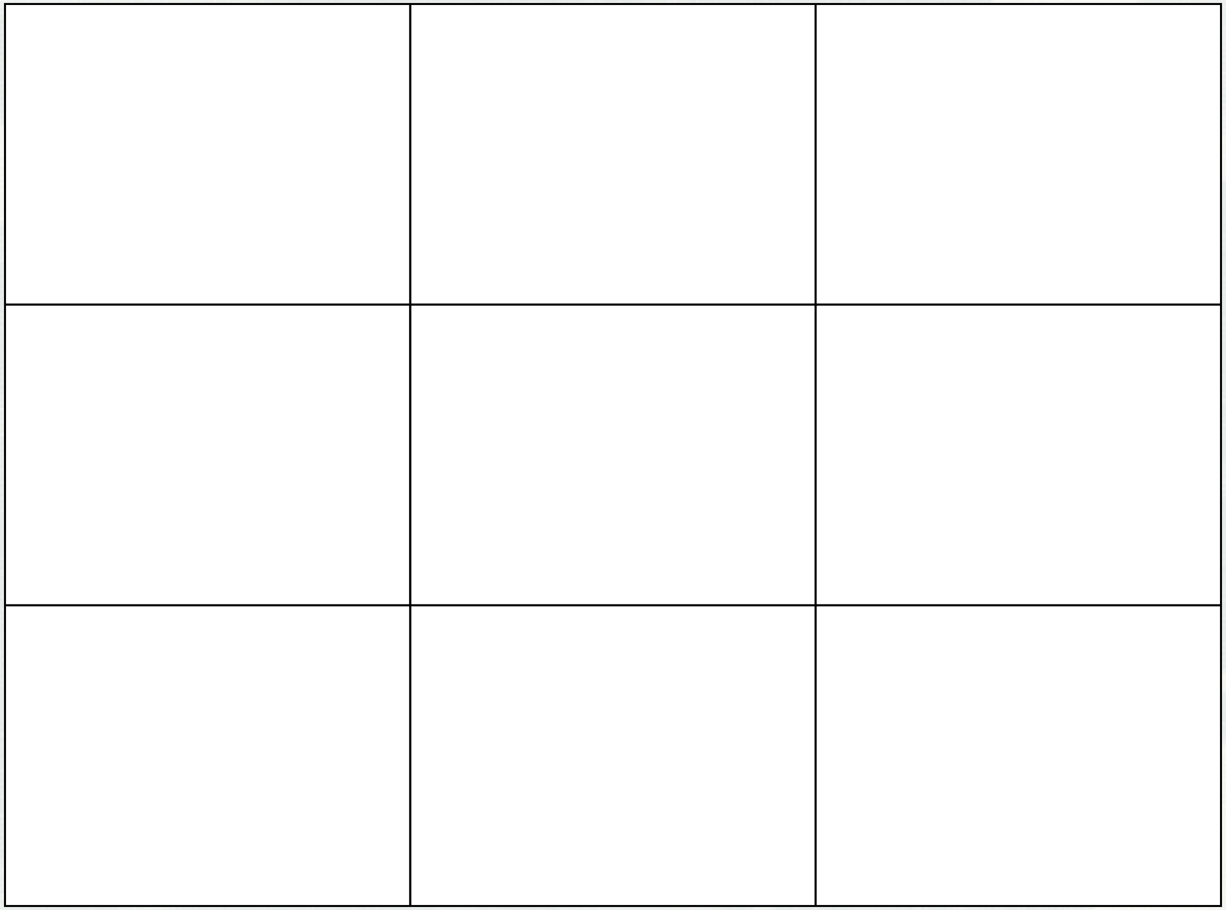
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)

LBM parallelization – streaming



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

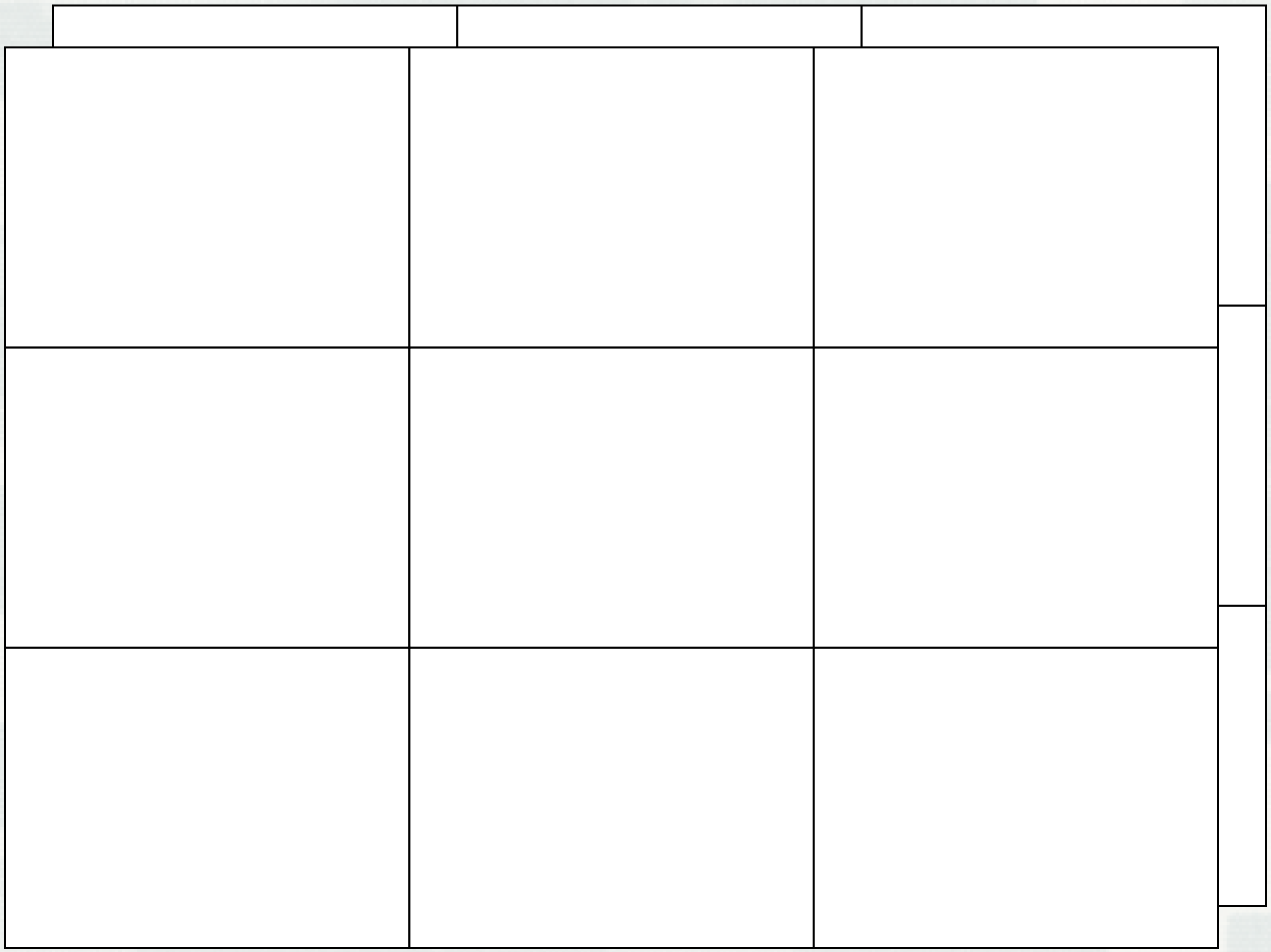
LBM parallelization – streaming



Direction

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

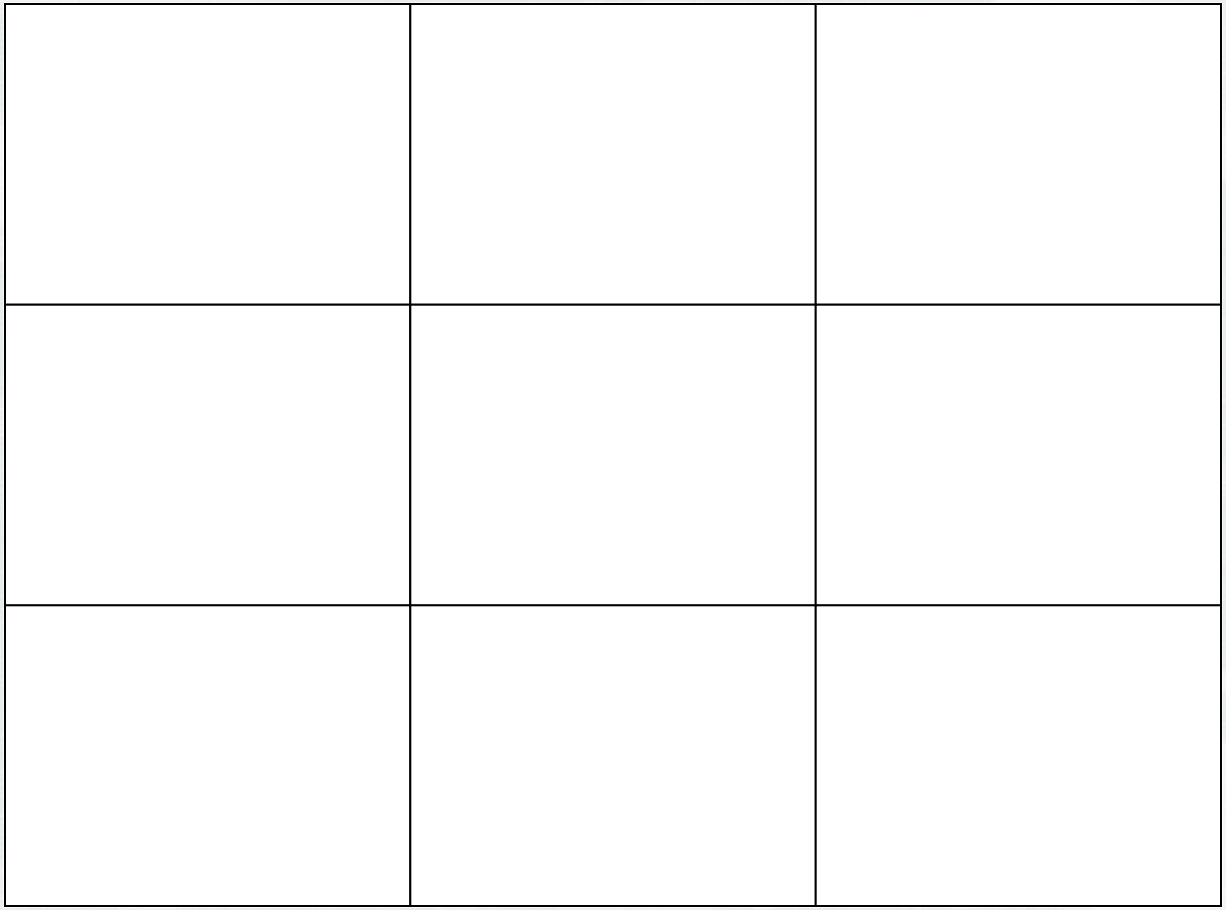
LBM parallelization – streaming



Direction

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

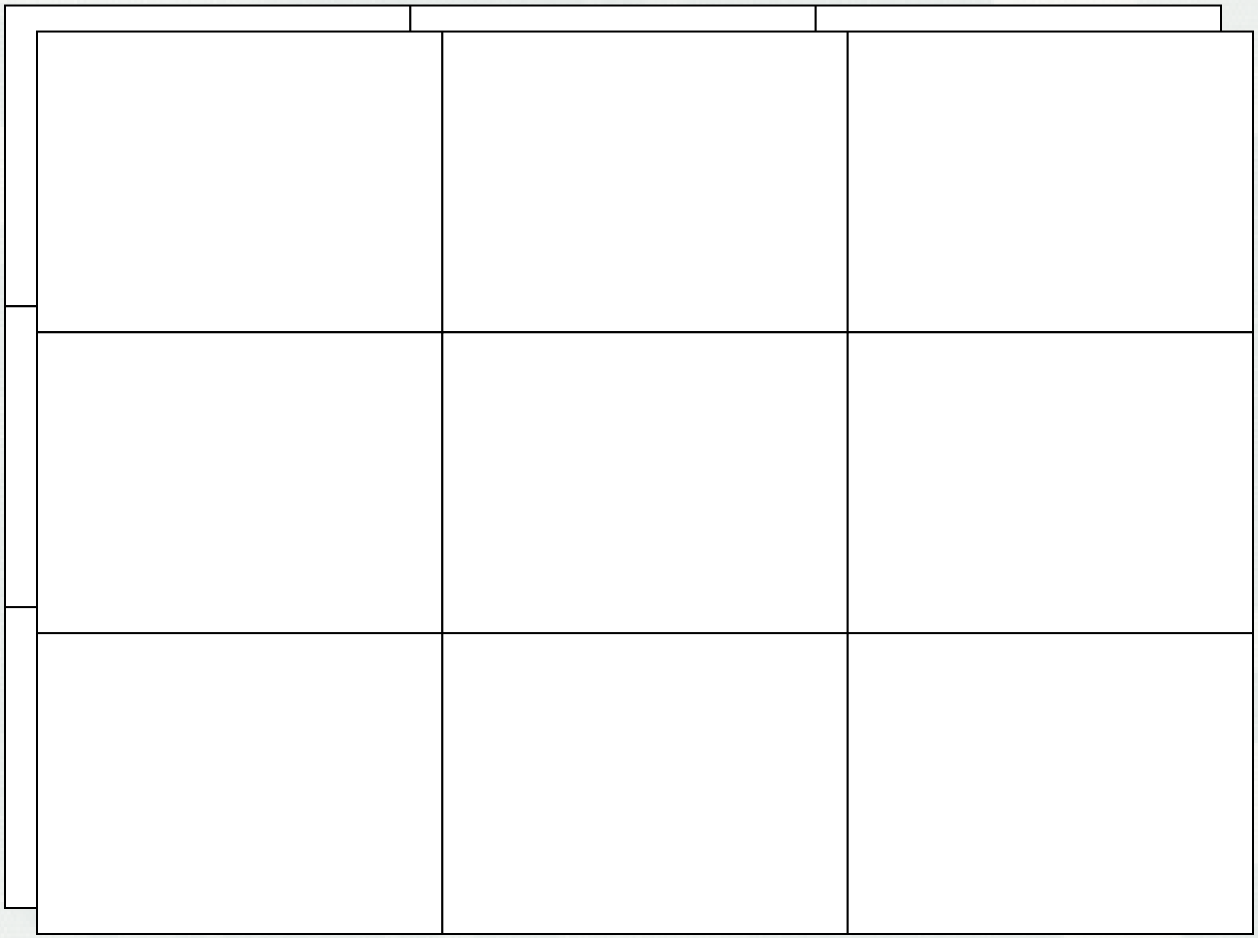
LBM parallelization – streaming



Direction

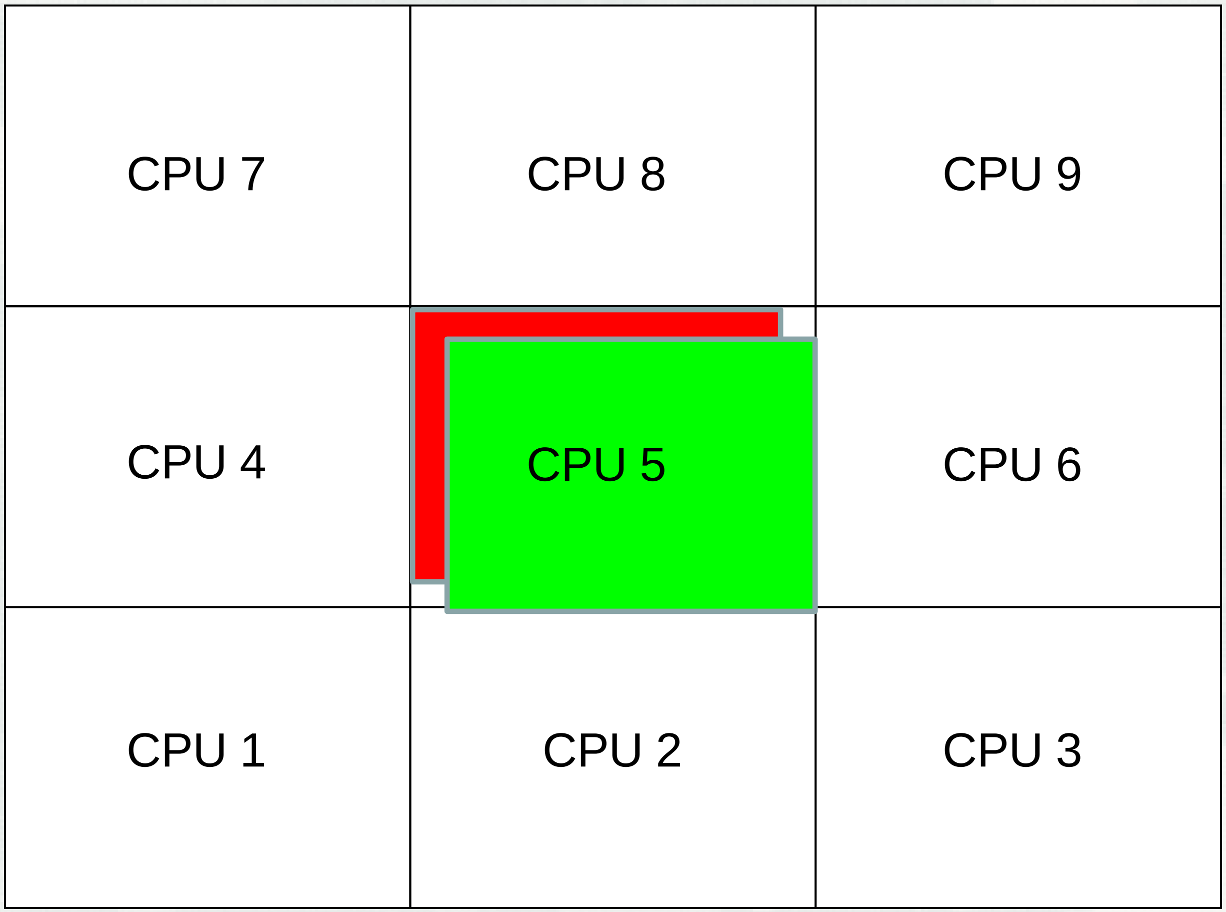
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)

LBM parallelization – streaming



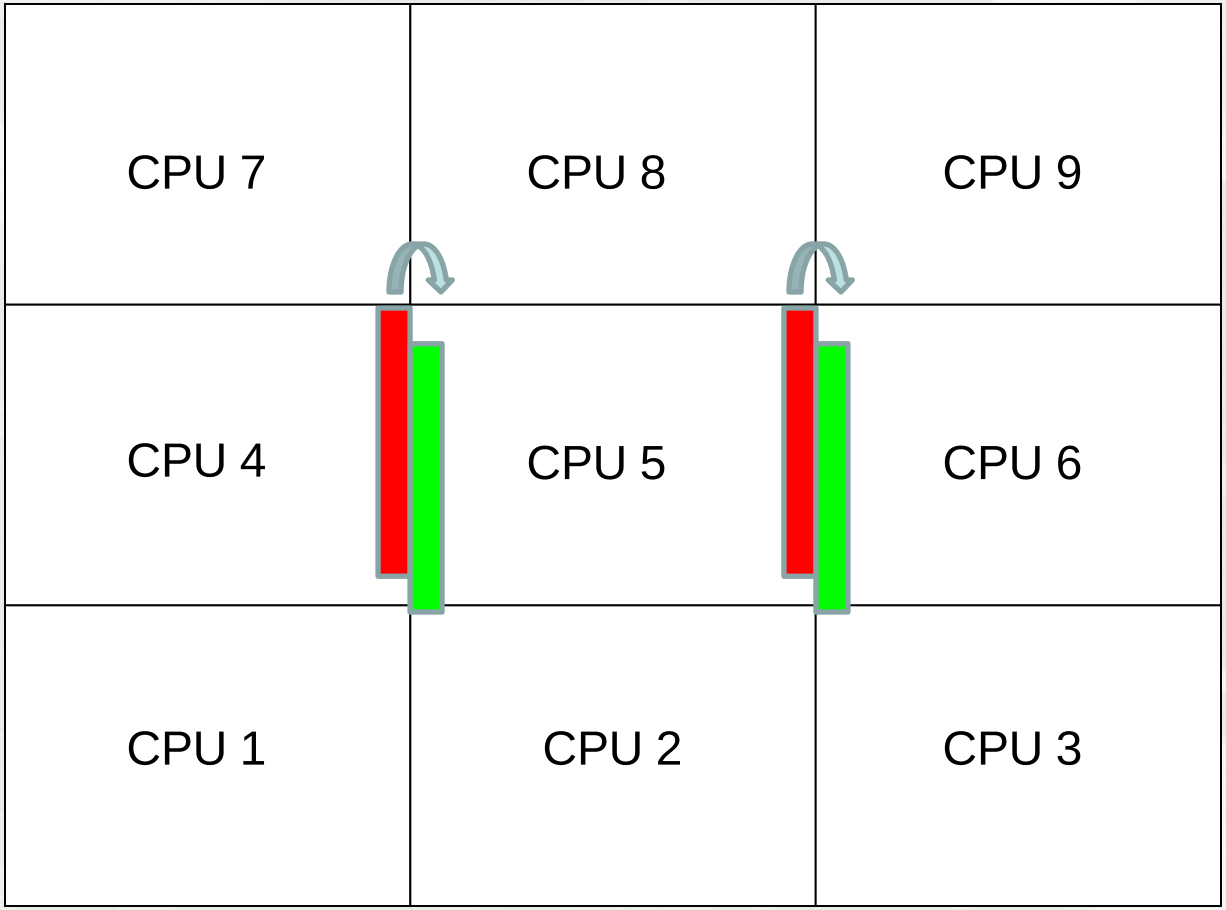
Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)

LBM parallelization – streaming



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

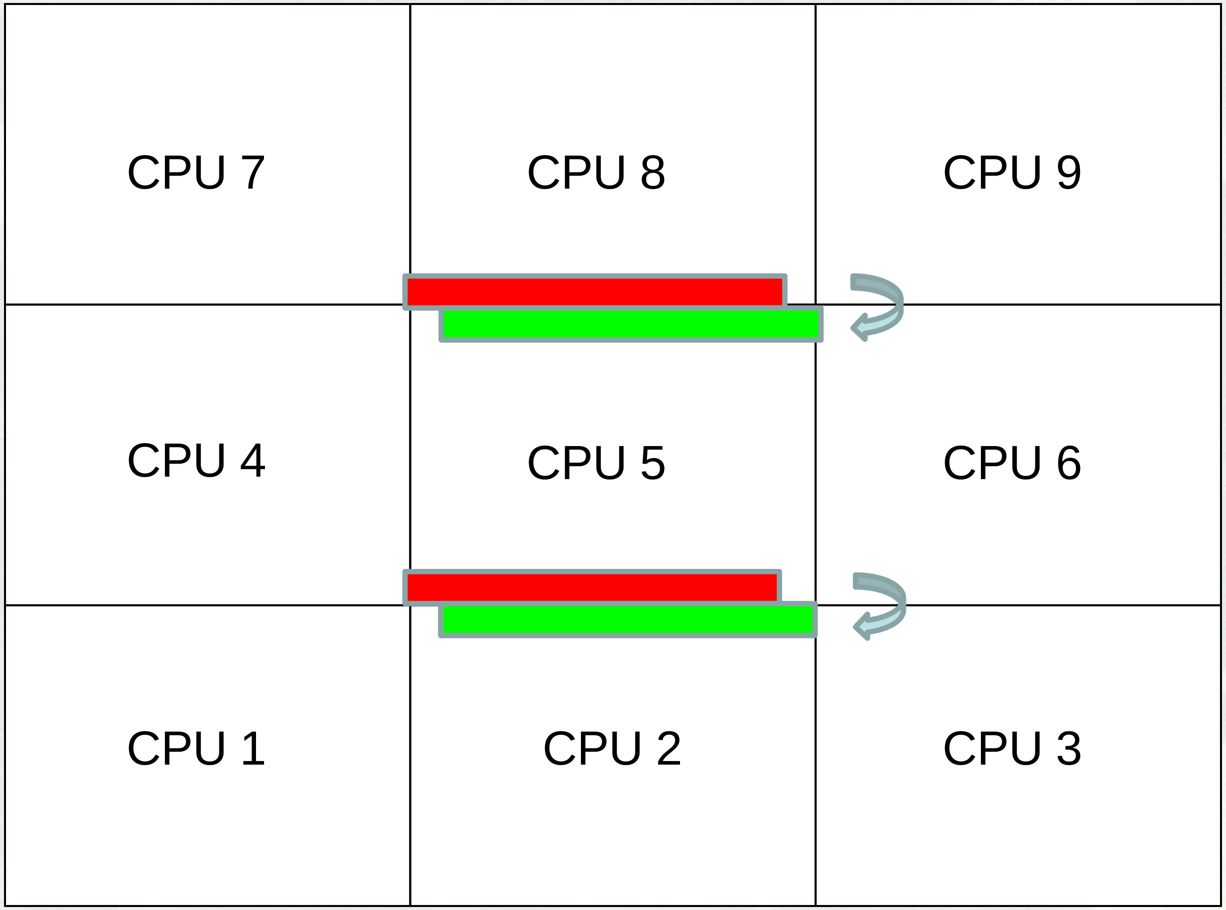
LBM parallelization – streaming



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

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

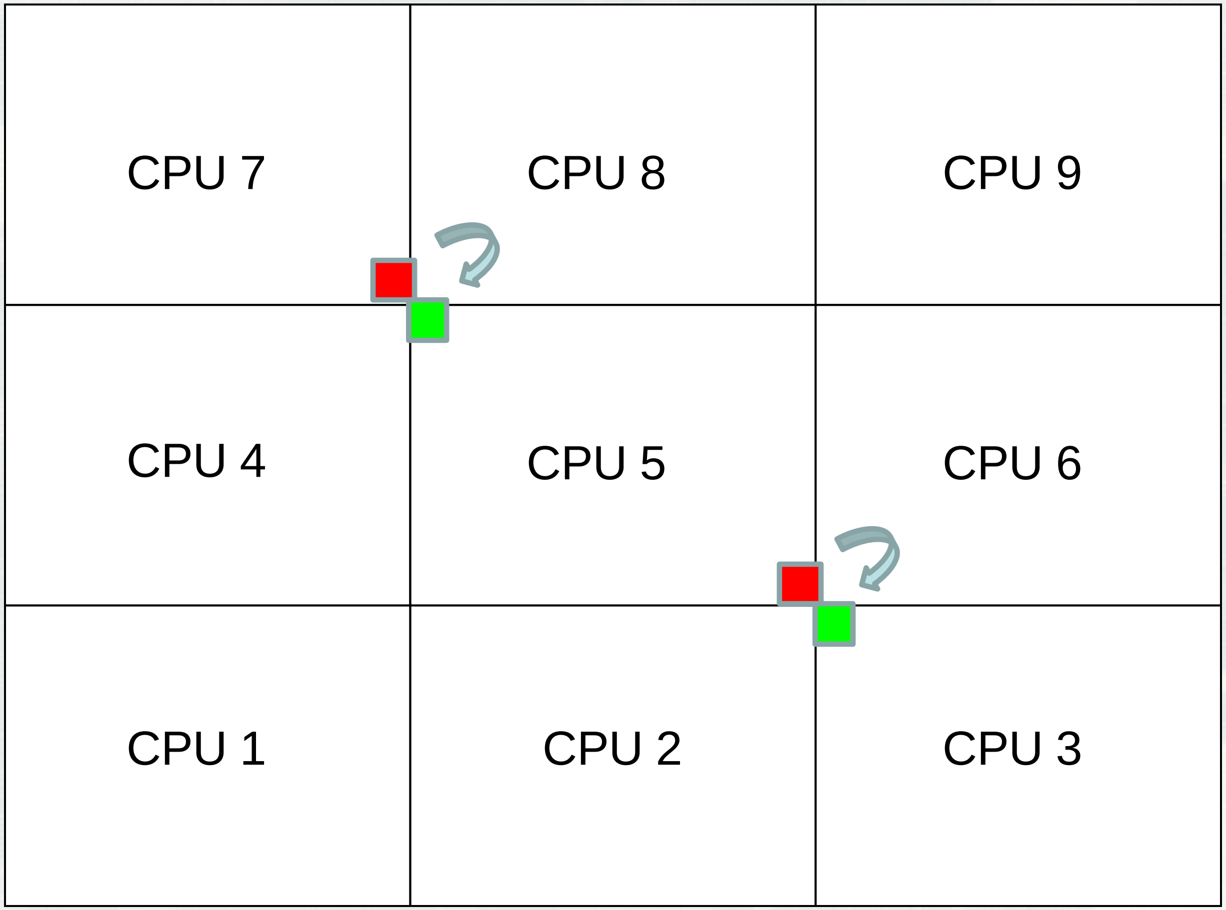
LBM parallelization – streaming



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

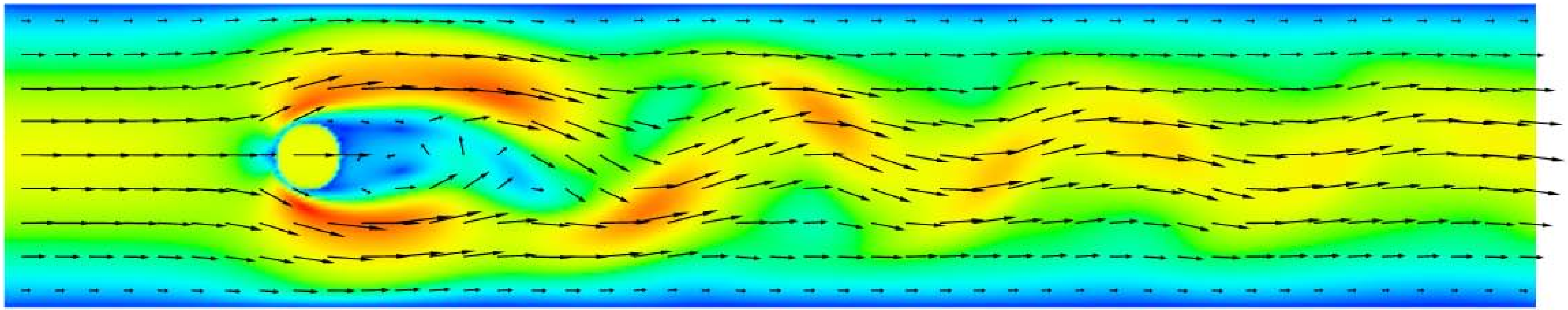


Direction
- horizontal (W, E)
- vertical (N, S)
- diagonal (NW, NE, SW, SE)

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

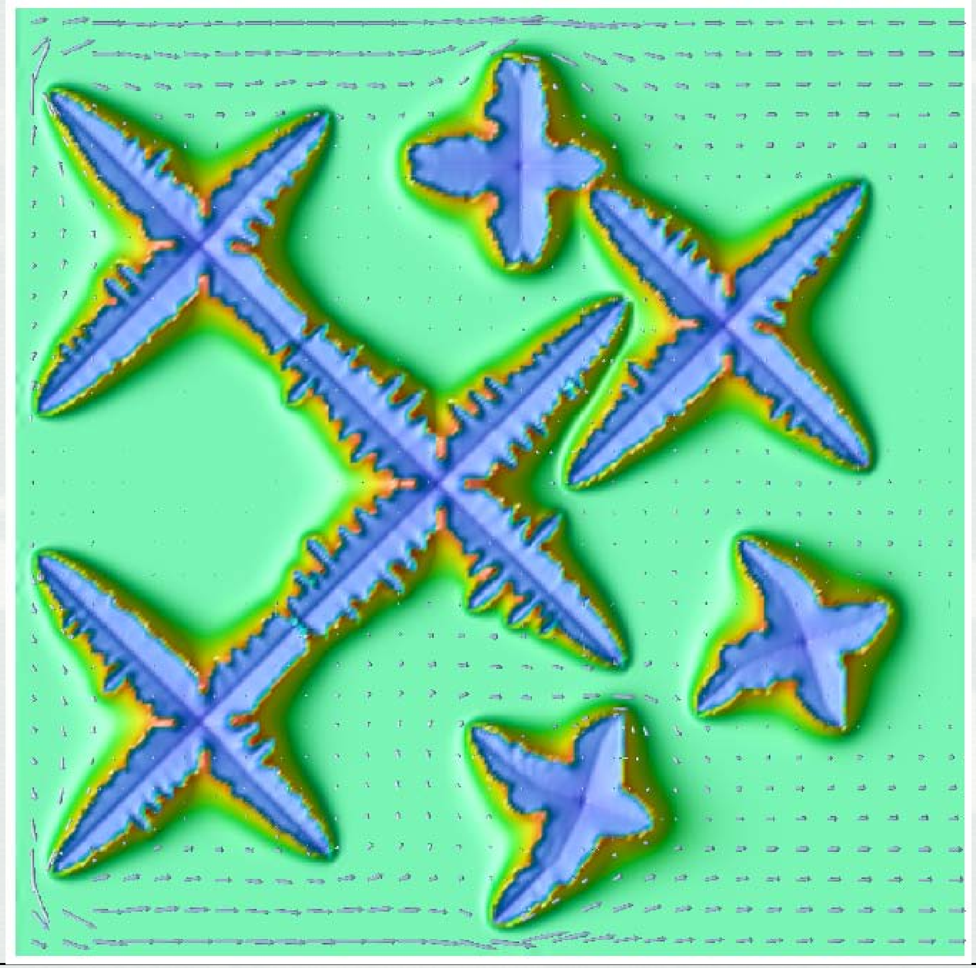
LBM parallelization – streaming

Street flow – example LBM problem:
velocity of flow

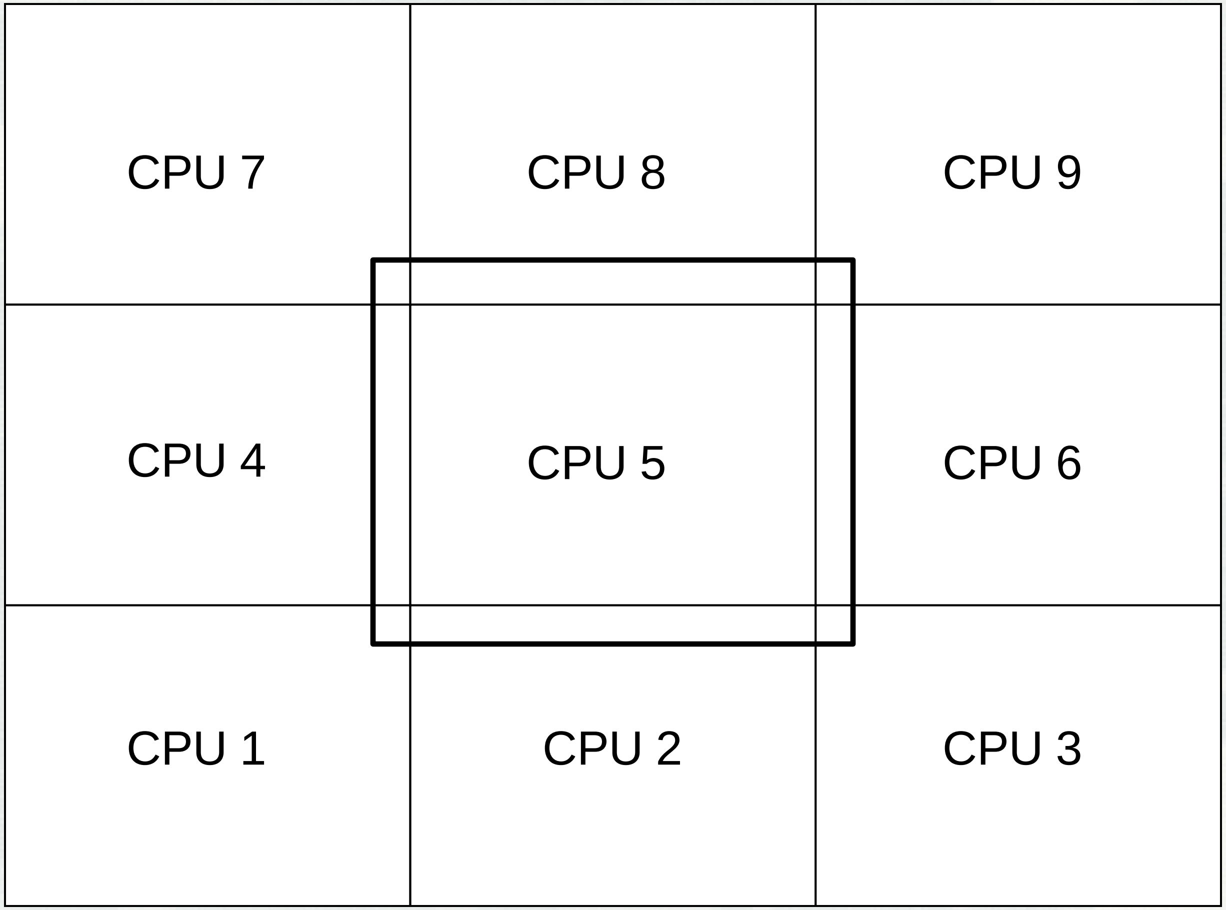


LBM parallelization

Dendrite growth in AlCu alloy upon cooling:
temperature, velocity of flow, and solute concentration

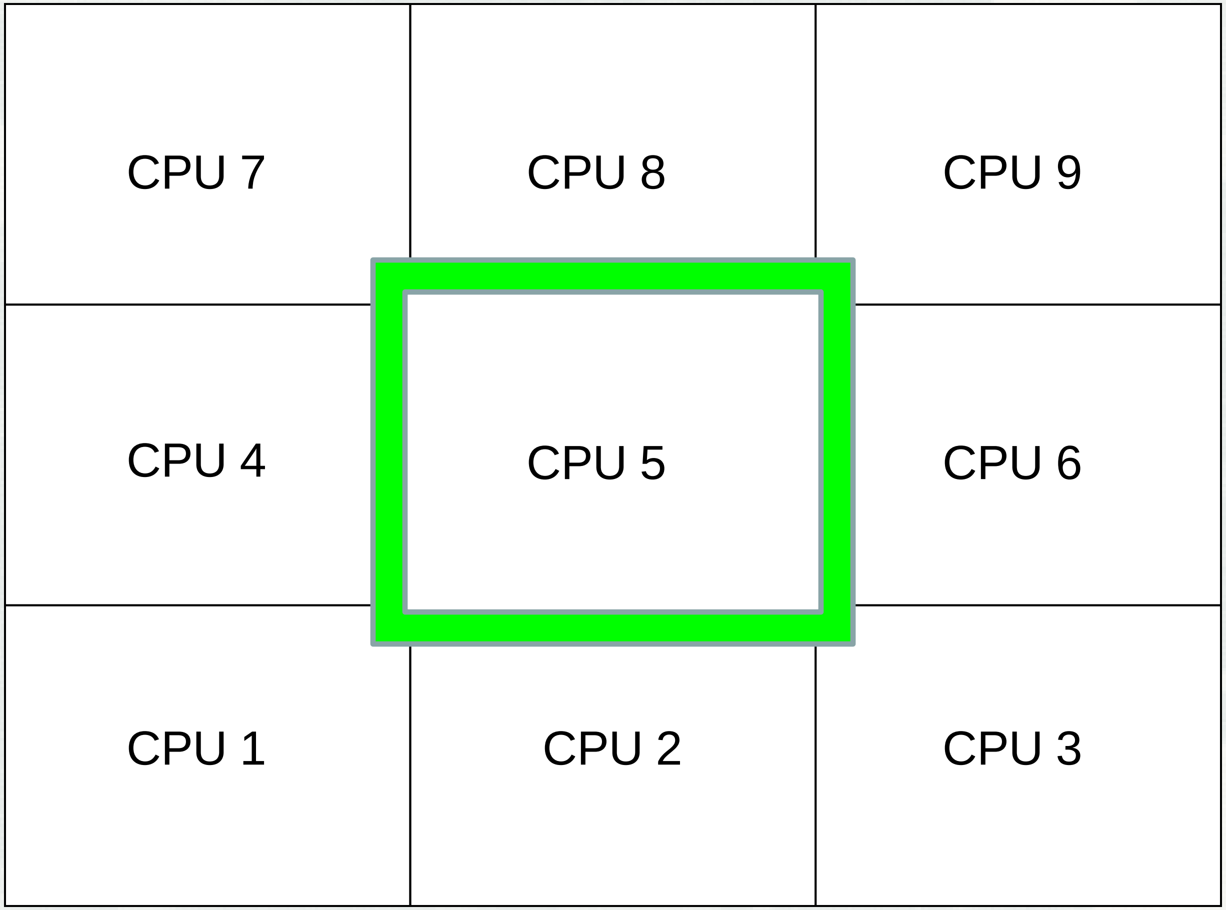


LBM parallelization



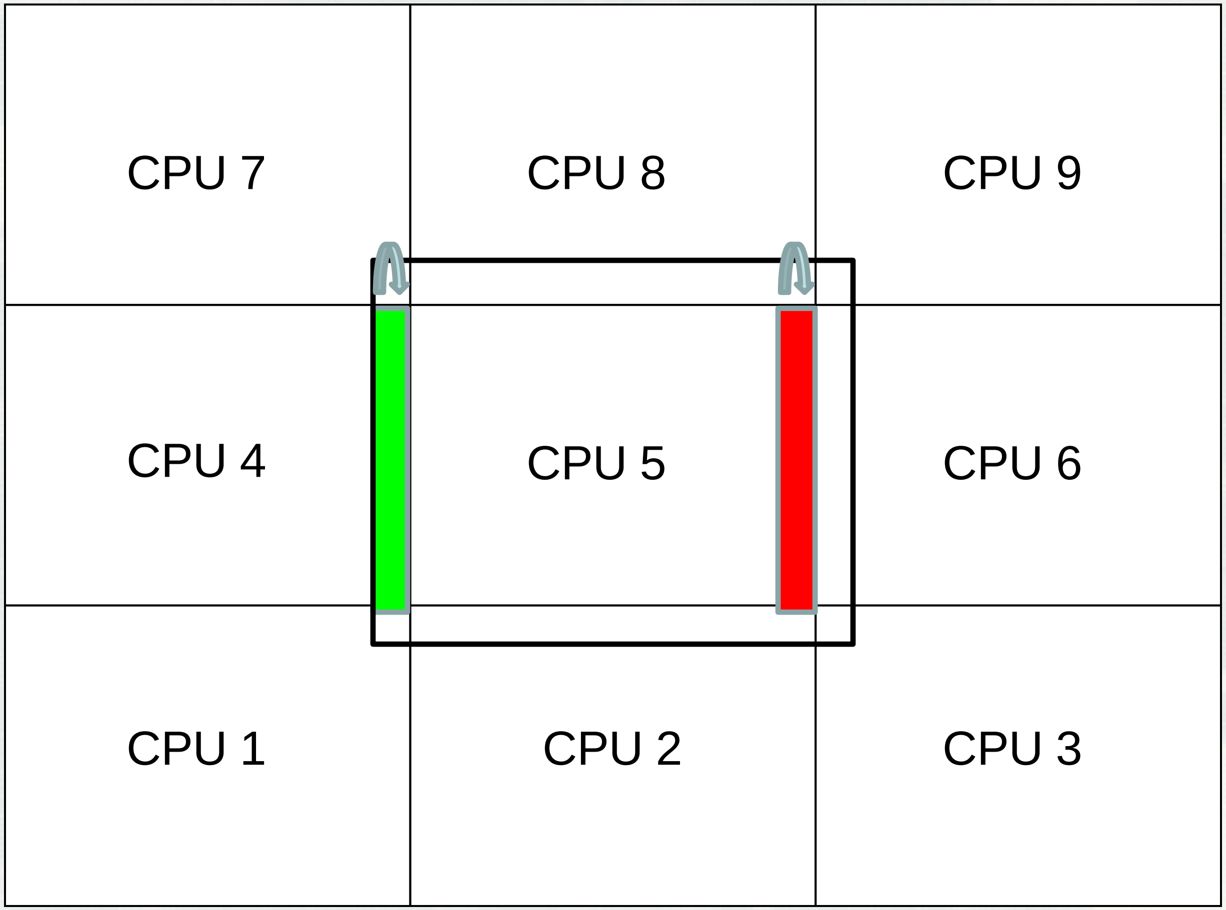
For dendrite growth, information from neighboring nodes is needed to update local node value

LBM parallelization – ghost nodes



Populate ghost nodes after each local update

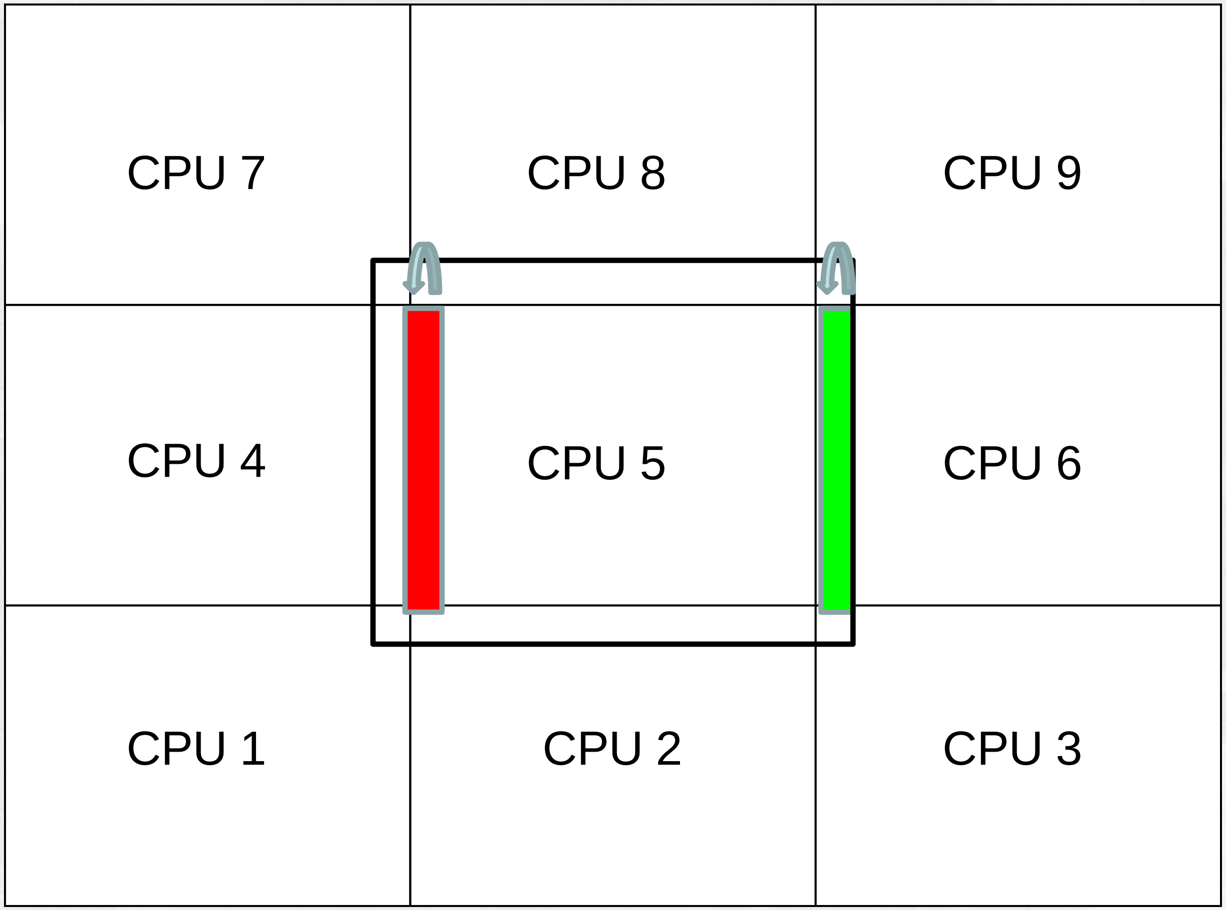
LBM parallelization – ghost nodes



Populate ghost nodes after each local update
east

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

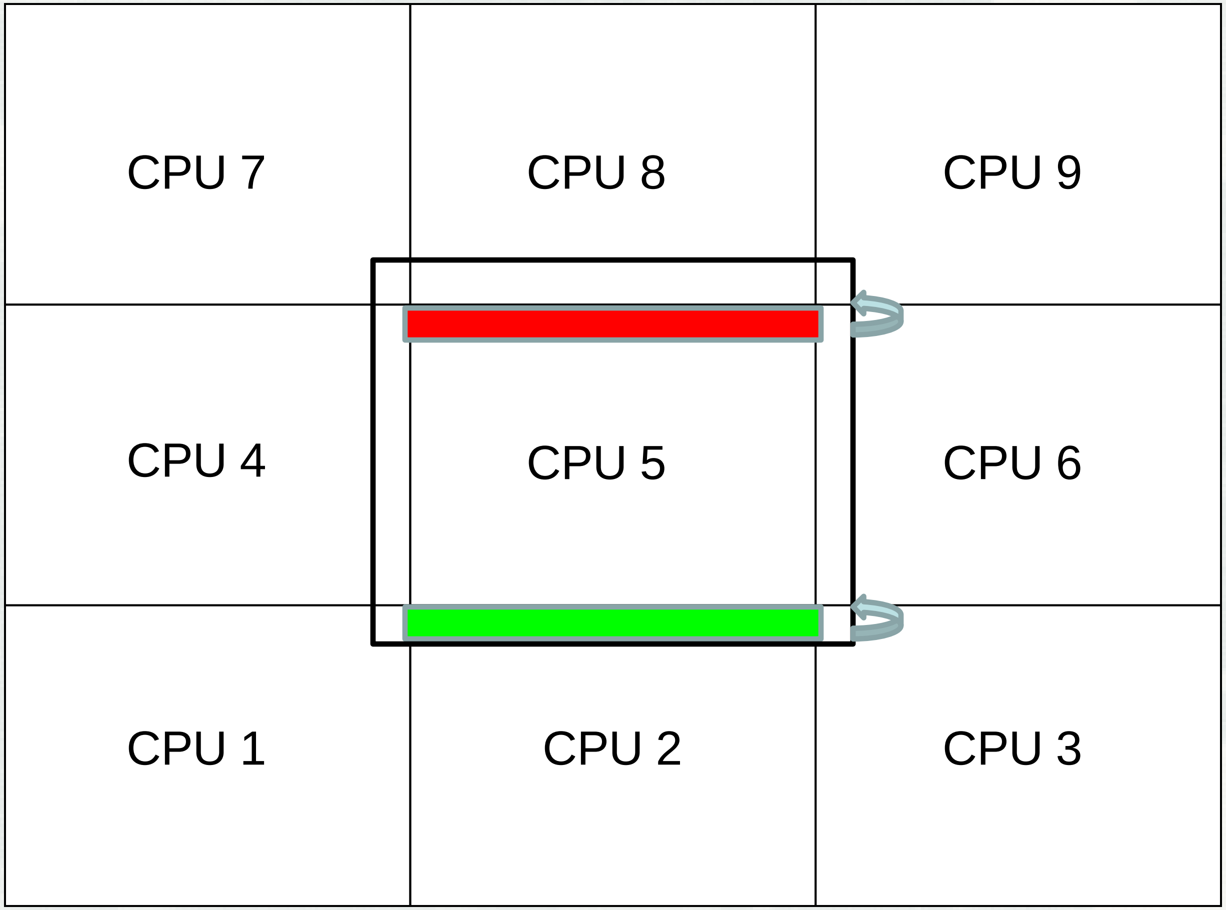
LBM parallelization – ghost nodes



Populate ghost nodes after each local update west

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

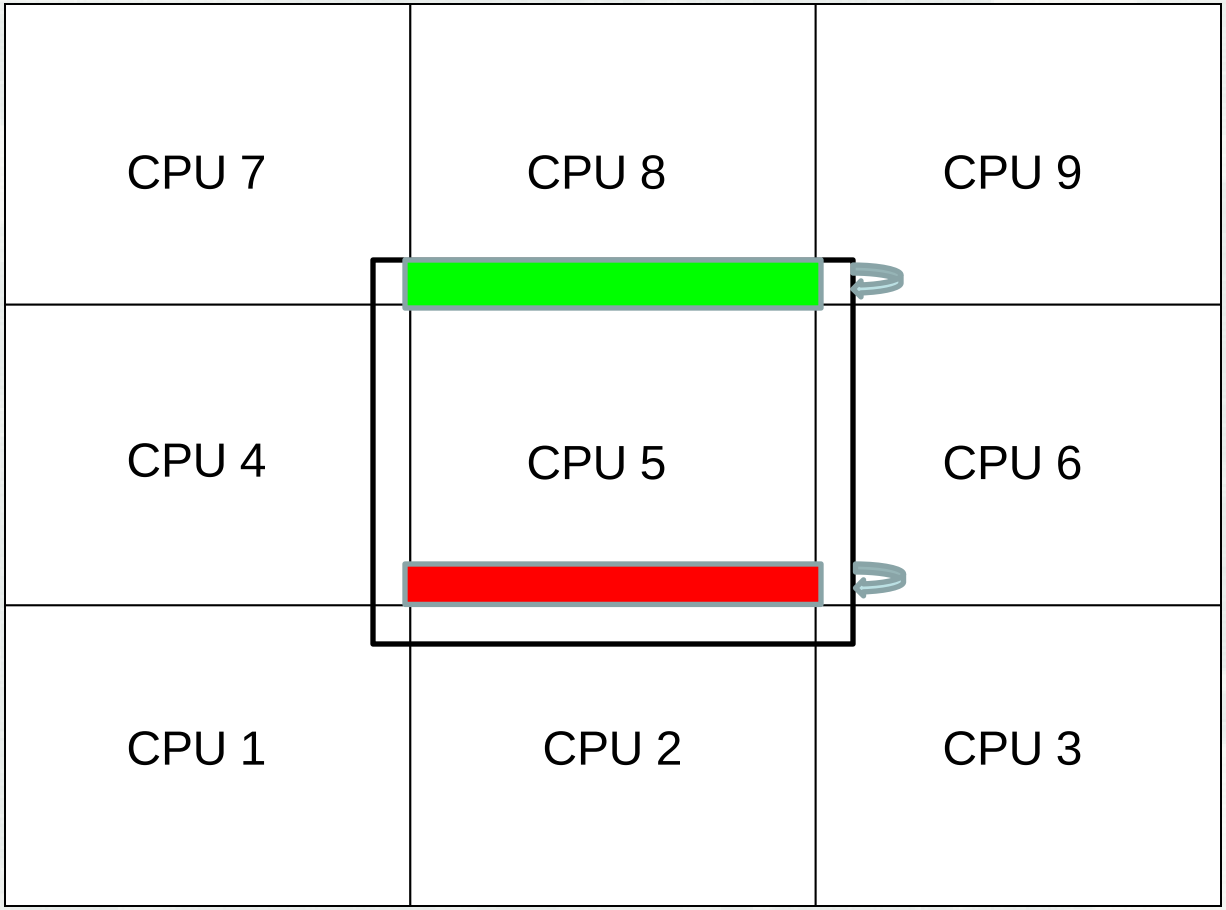
LBM parallelization – ghost nodes



Populate ghost nodes after each local update
north

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

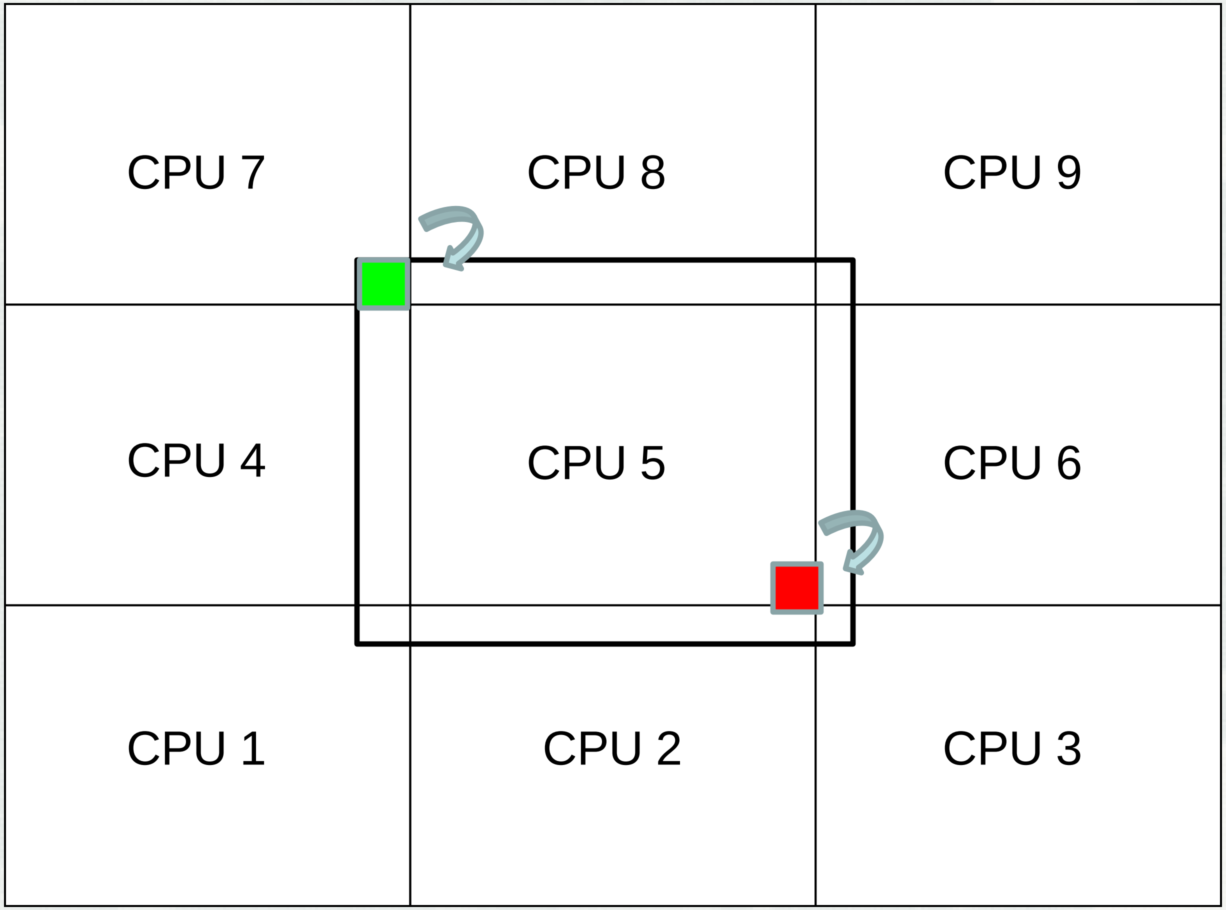
LBM parallelization – ghost nodes



Populate ghost nodes after each local update south

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

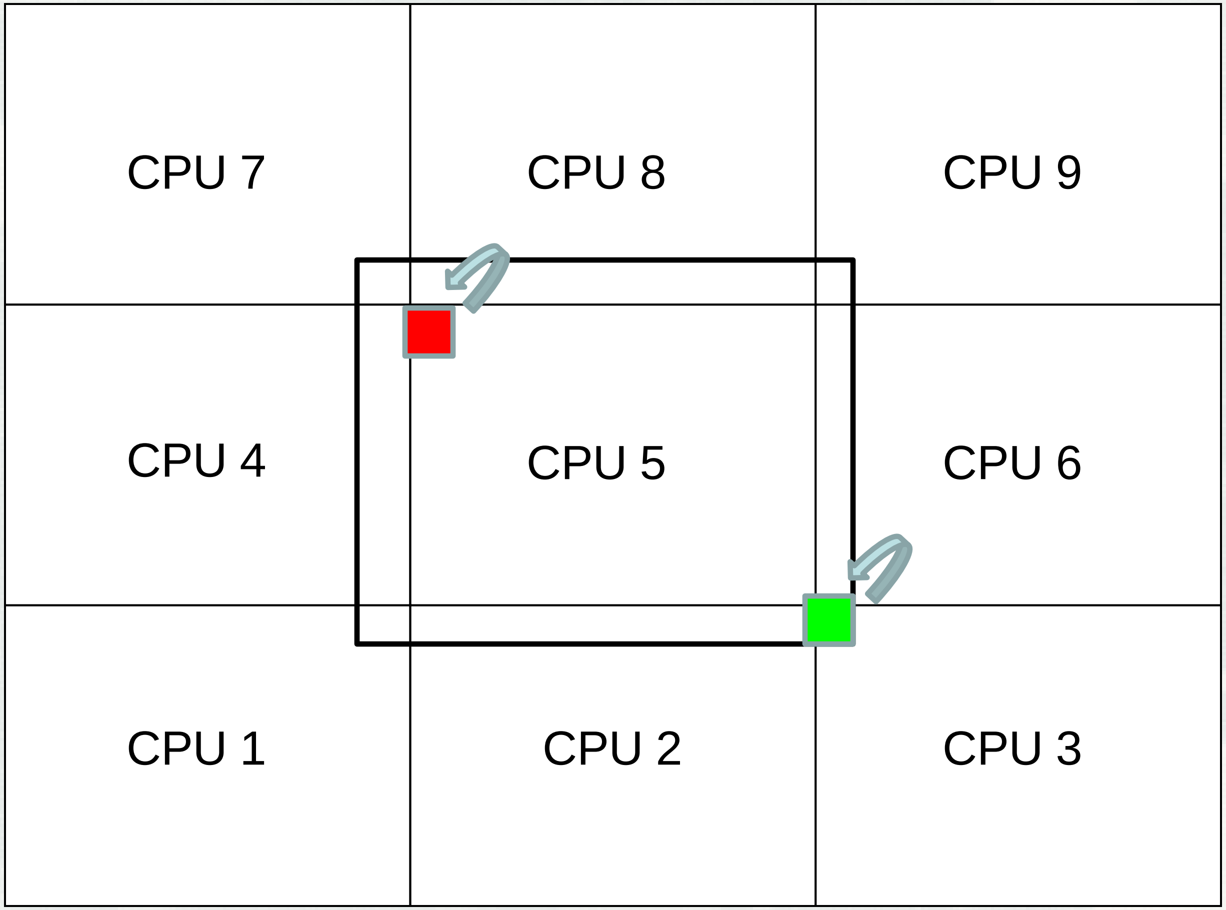
LBM parallelization – ghost nodes



Populate ghost nodes after each local update south-east

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

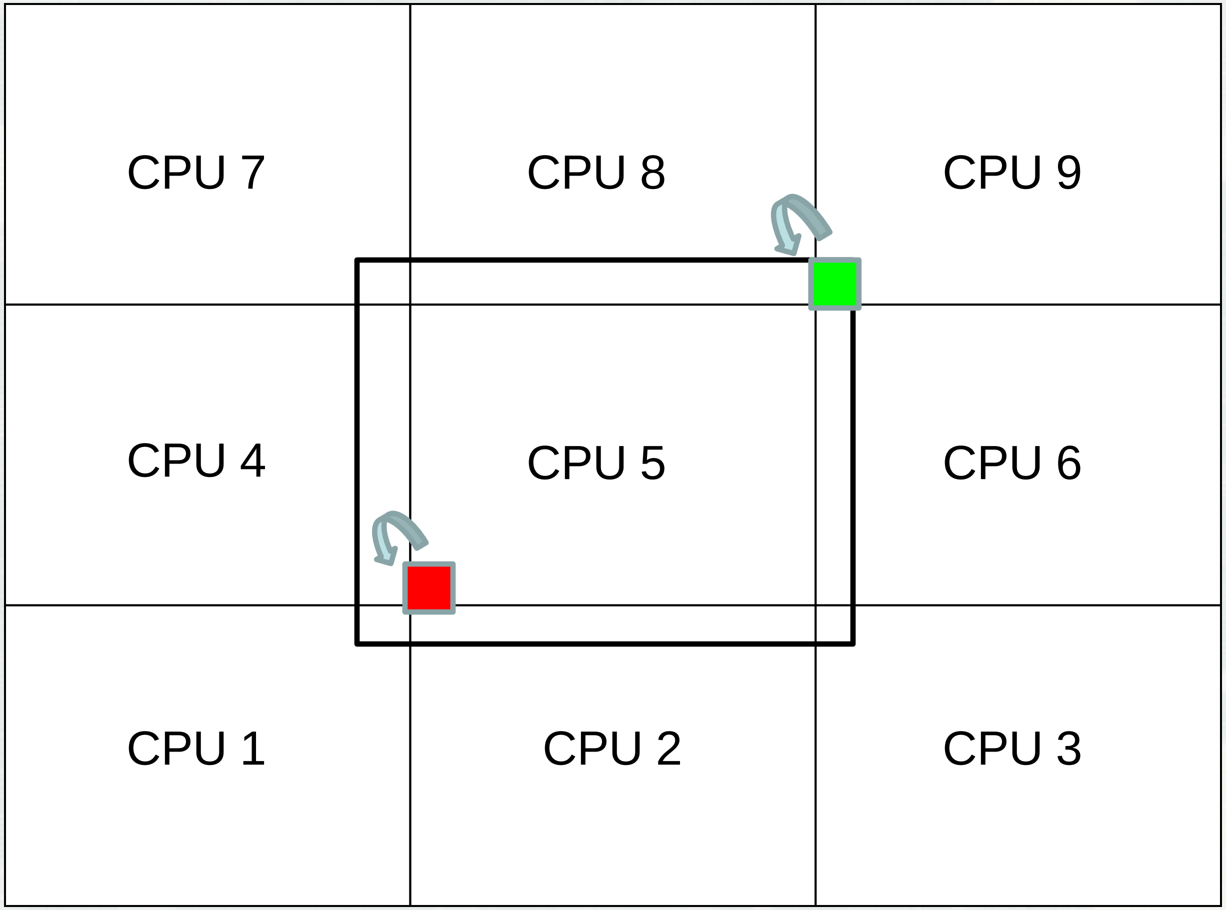
LBM parallelization – ghost nodes



Populate ghost nodes after each local update north-west

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

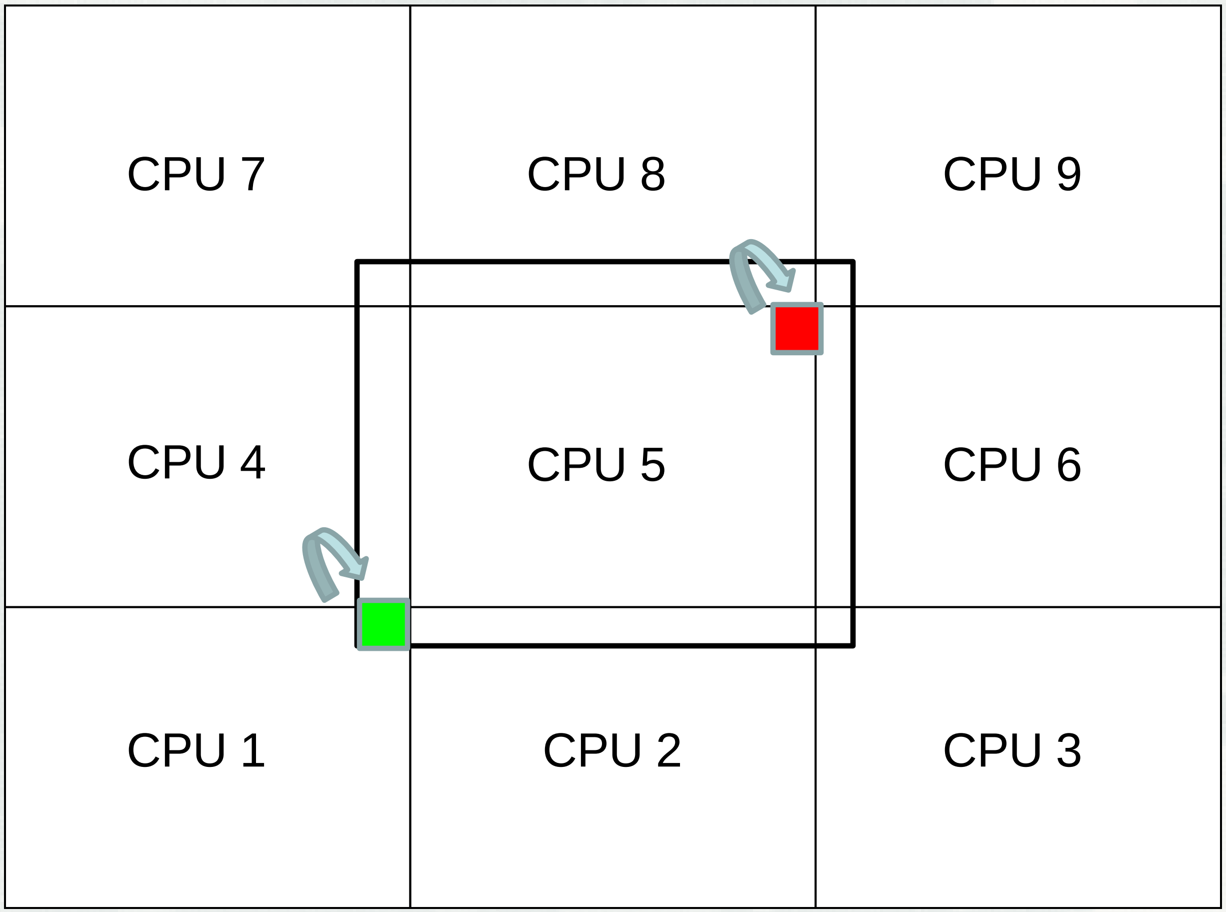
LBM parallelization – ghost nodes



Populate ghost nodes after each local update south-west

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

LBM parallelization – ghost nodes

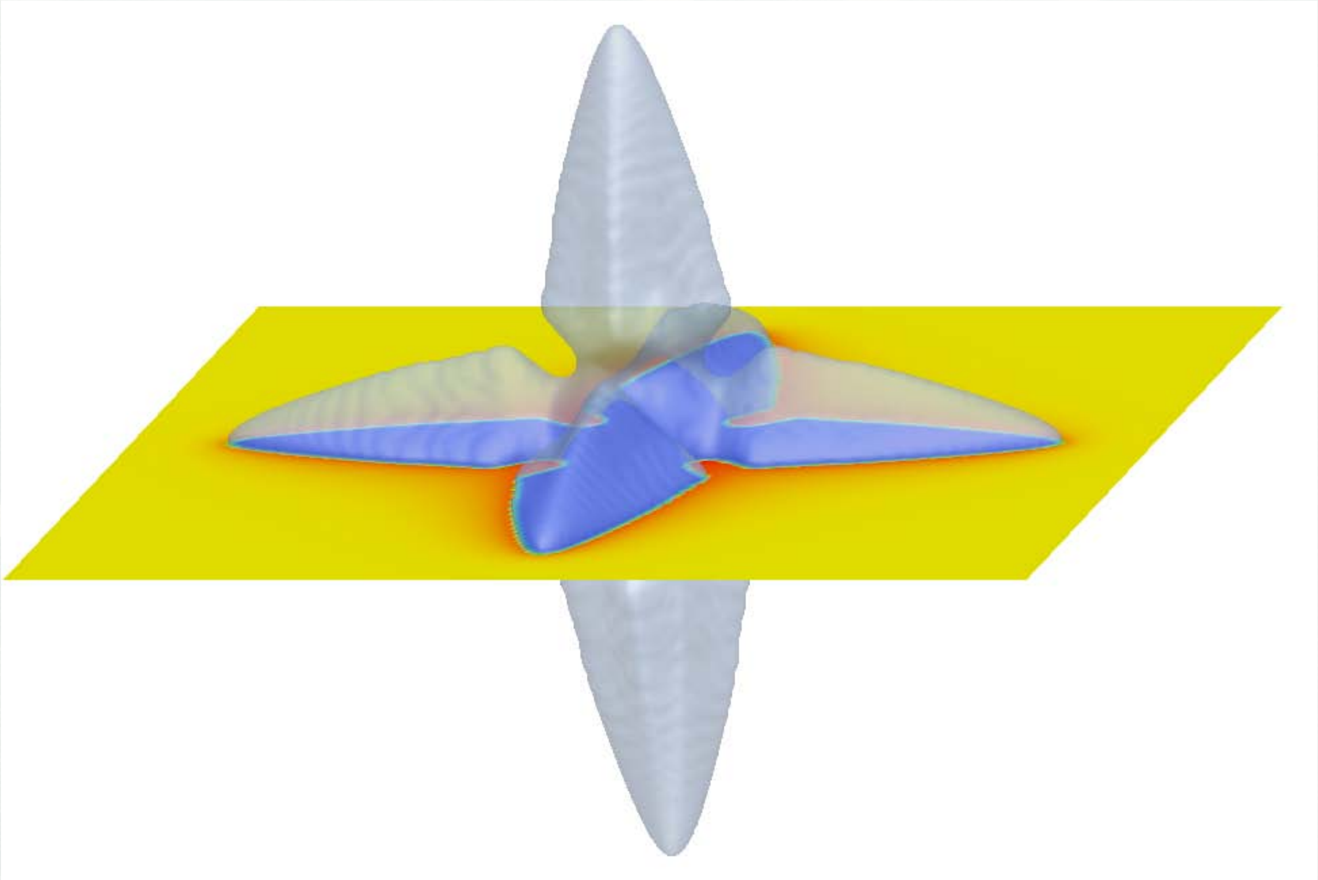


Populate ghost nodes after each local update north-east

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

LBM parallelization

3D Dendrite growth in AlCu alloy upon cooling: temperature, fluid flow, and solute concentration



Conclusions

Demonstrated an improved prediction of velocity profile from charge density with non-constant viscosity estimated from MD simulations

Revealed the dependence of the flow on surface charge density, distribution, and ionic concentrations

Parallelized 2D Lattice-Boltzmann code including dendrite growth

Implement MD-LBM / MD-DEM coupling