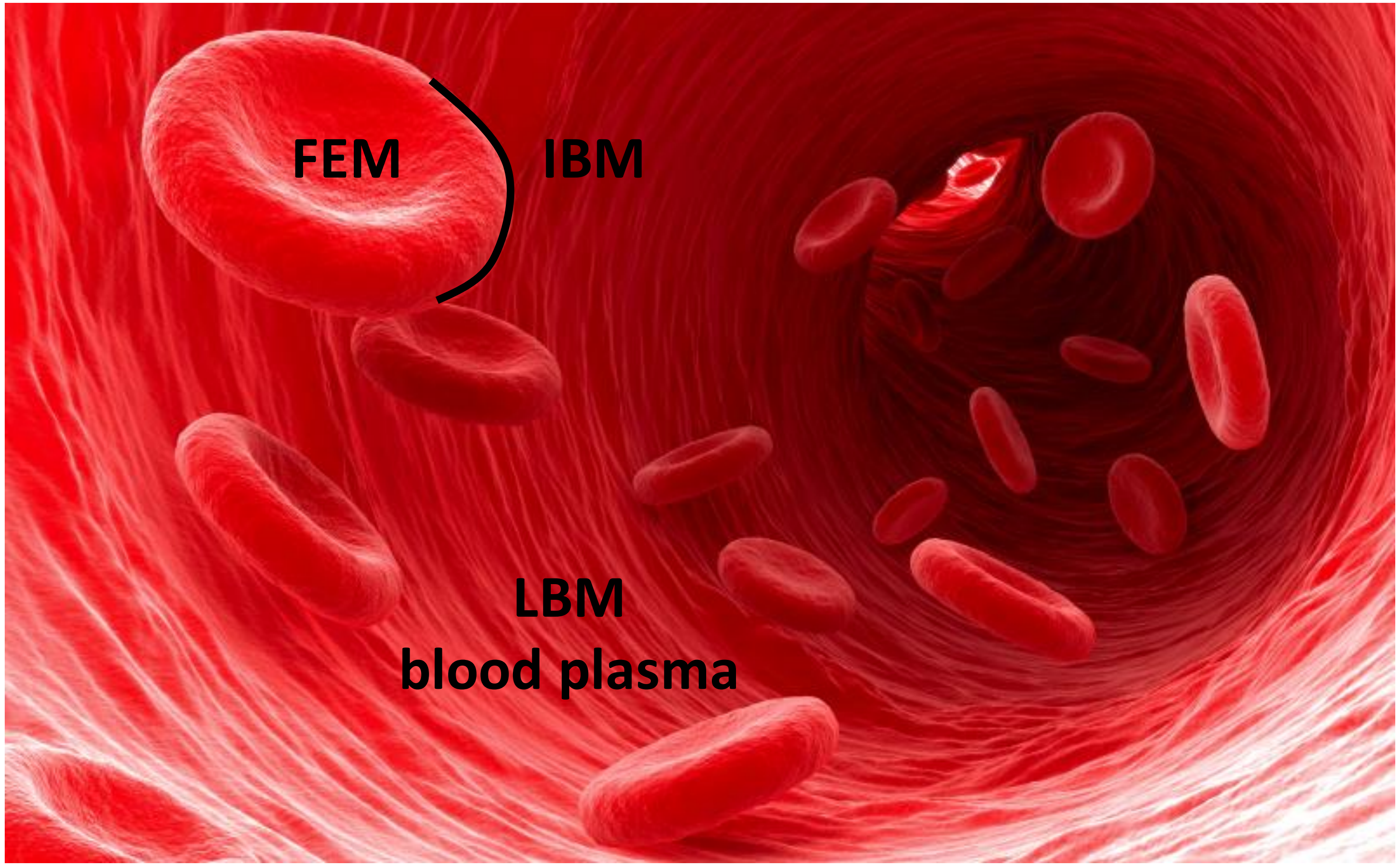
High fidelity and fast simulations of deformable blood cells using a combined Finite Element Immersed Boundary Lattice Boltzmann method (FE-IB-LBM)

Palabos-npFEM module

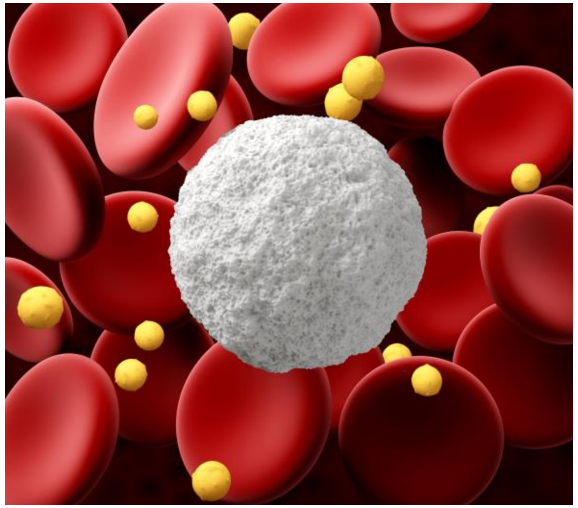
José Pedro de Santana Neto (This presentation and software implementation was created by Christos Kotsalos and all the credits are given to him.)

Scientific and Parallel Computing Group (SPC)

High fidelity and fast simulations of **deformable blood cells** using a combined **Finite Element Immersed Boundary Lattice Boltzmann** method (FE-IB-LBM)



- FEM for the blood cells
- LBM for the blood plasma
- IBM for the Fluid-Solid Interaction



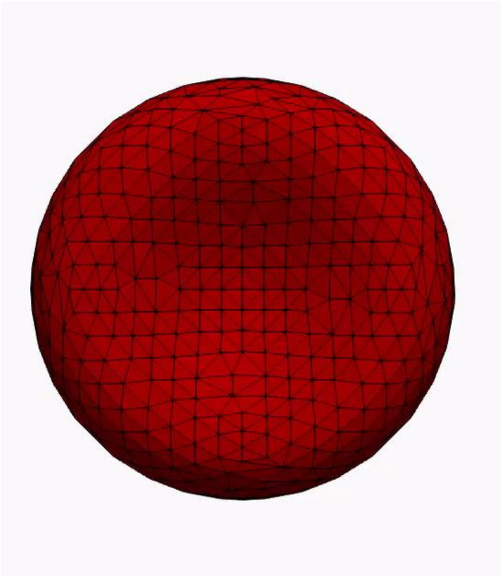
Numerical methods:

- **Lattice Boltzmann** for the blood plasma (Palabos in **C++/MPI**)
- **Immersed boundary** for the coupling (Palabos in **C++**)
- Mass-lumped **FEM** for the deformable bodies (developed in **C++/CUDA**)

Palabos-npFEM module for cellular blood flow simulations
(same principles apply to problems of other domains)

PALABOS

Mass-lumped FEM (**nodal projective FEM**)



Implicit Euler time integration:
update rule

Newton's 2nd law per vertex

$$\boldsymbol{F}_{int}(\boldsymbol{x}_{n+1}) + \boldsymbol{F}_{ext} - \mathbf{C}\boldsymbol{v}_{n+1} = \mathrm{M}\frac{\boldsymbol{v}_{n+1} - \boldsymbol{v}_n}{h}$$

$$\boldsymbol{v}_{n+1} = \frac{\boldsymbol{x}_{n+1} - \boldsymbol{x}_n}{h}$$

Subscripts n/n+1 refer to time t and t+1

$$\boldsymbol{F}_{int}(\boldsymbol{x}) = -\sum_i \nabla E_i(\mathbf{x})$$

E is the potential energy stored in the body

Variational Implicit Euler formulation

$$\min_{\boldsymbol{x}_{n+1}} \quad \frac{1}{2h^2} \left\| \widetilde{\mathbf{M}}^{\frac{1}{2}} (\boldsymbol{x}_{n+1} - \boldsymbol{y}_n) \right\|_F^2 + \sum_i E_i(\boldsymbol{x}_{n+1})$$

Mass-lumped FEM (**npFEM**)

$$\min_{\boldsymbol{x}_{n+1}} \; g(\boldsymbol{x}_{n+1}) = \frac{1}{2h^2} \left\| \widetilde{\mathbf{M}}^{\frac{1}{2}} (\boldsymbol{x}_{n+1} - \boldsymbol{y}_n) \right\|_F^2 + \sum_i E_i(\boldsymbol{x}_{n+1})$$

4 different potential energies to describe a blood cell:

- Area Conservation
- Global Volume Conservation
- Bending rigidity
- Material model (modified Skalak)

Quasi-Newton: $\boldsymbol{x}_{n+1}^{k+1} = \boldsymbol{x}_{n+1}^{k} - \alpha \widetilde{\mathbf{H}}^{-1} \nabla g(\boldsymbol{x}_{n+1}^{k})$

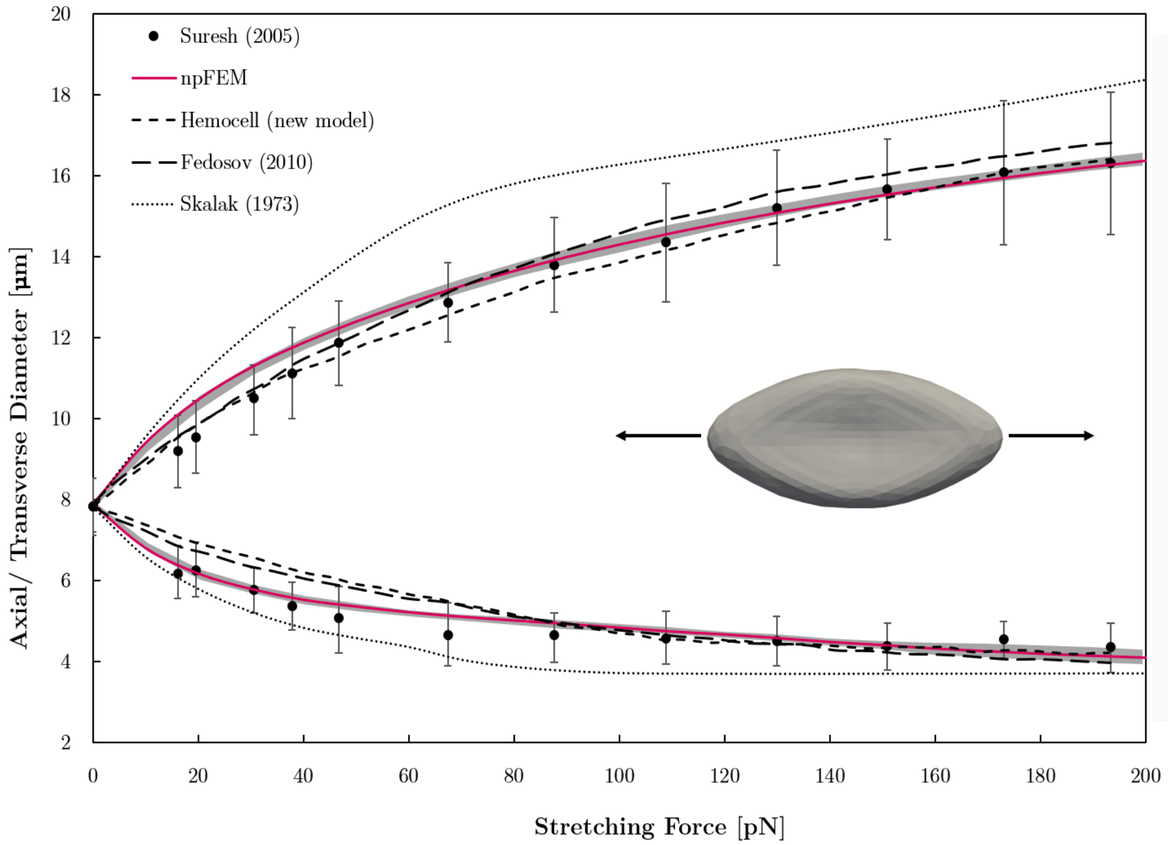
For more details:

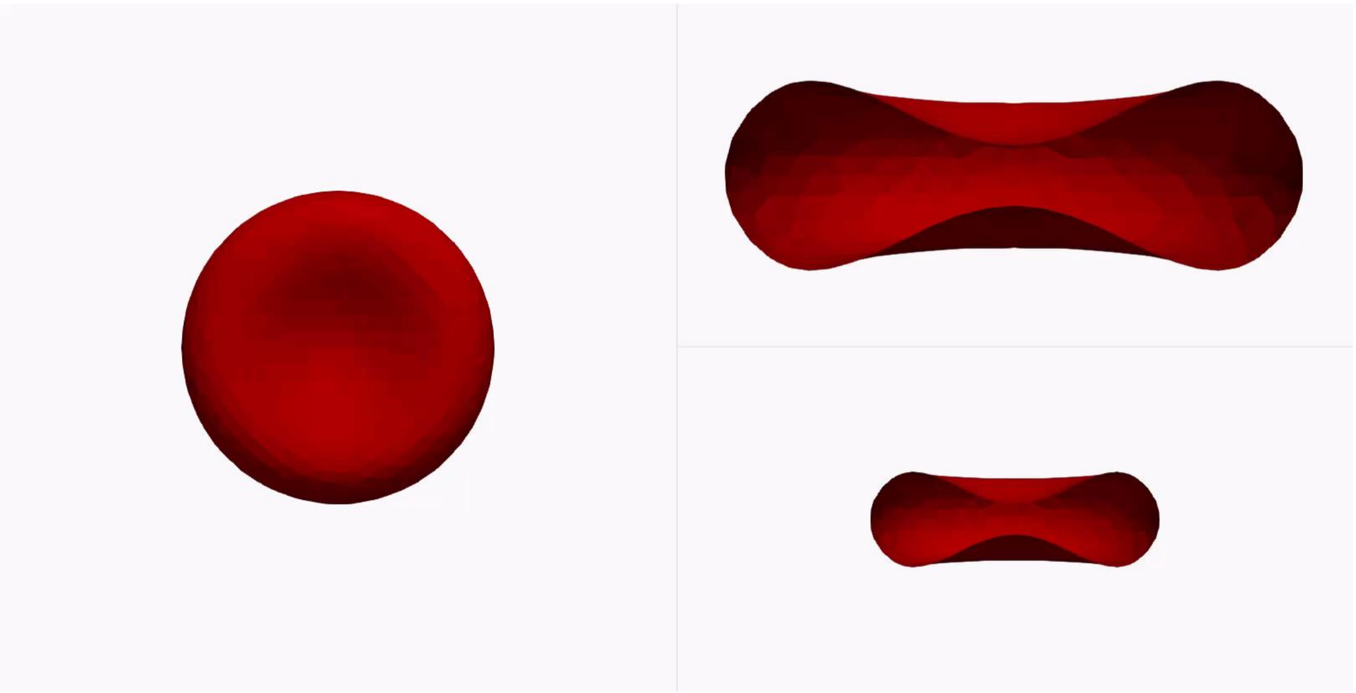
Bridging the computational gap between mesoscopic and continuum modeling of red blood cells for fully resolved blood flow

Journal of Computational Physics 2019

https://doi.org/10.1016/j.jcp.2019.108905

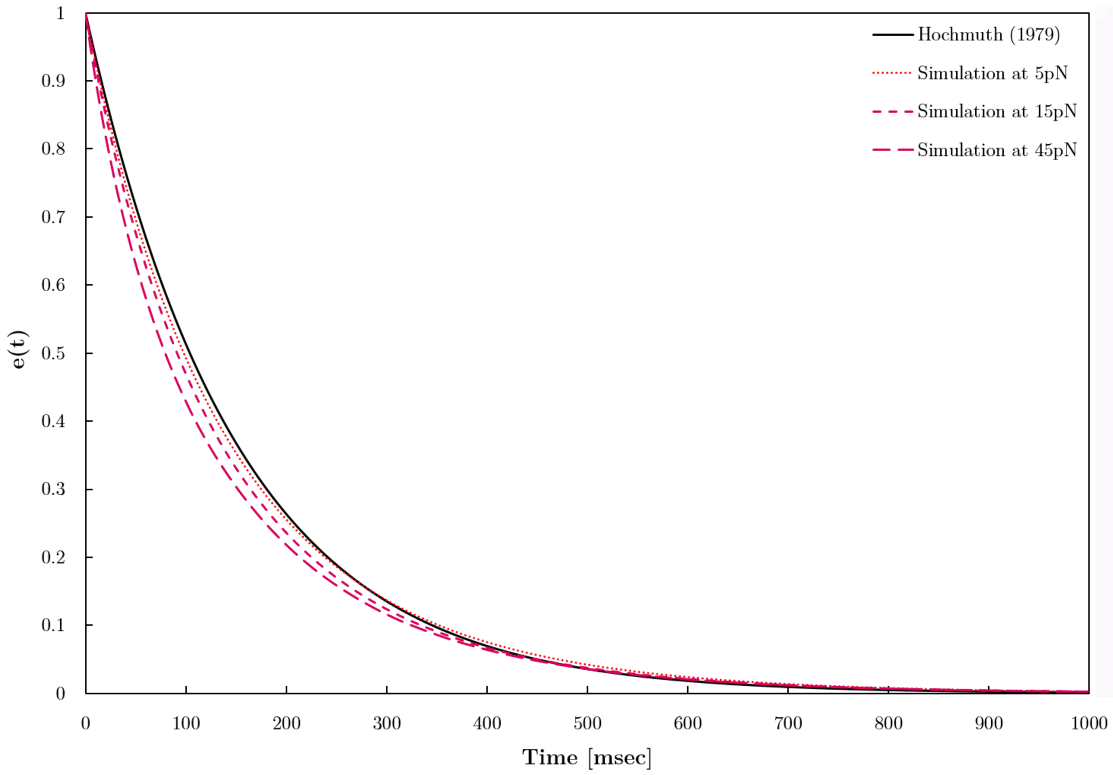
Stretching experiment

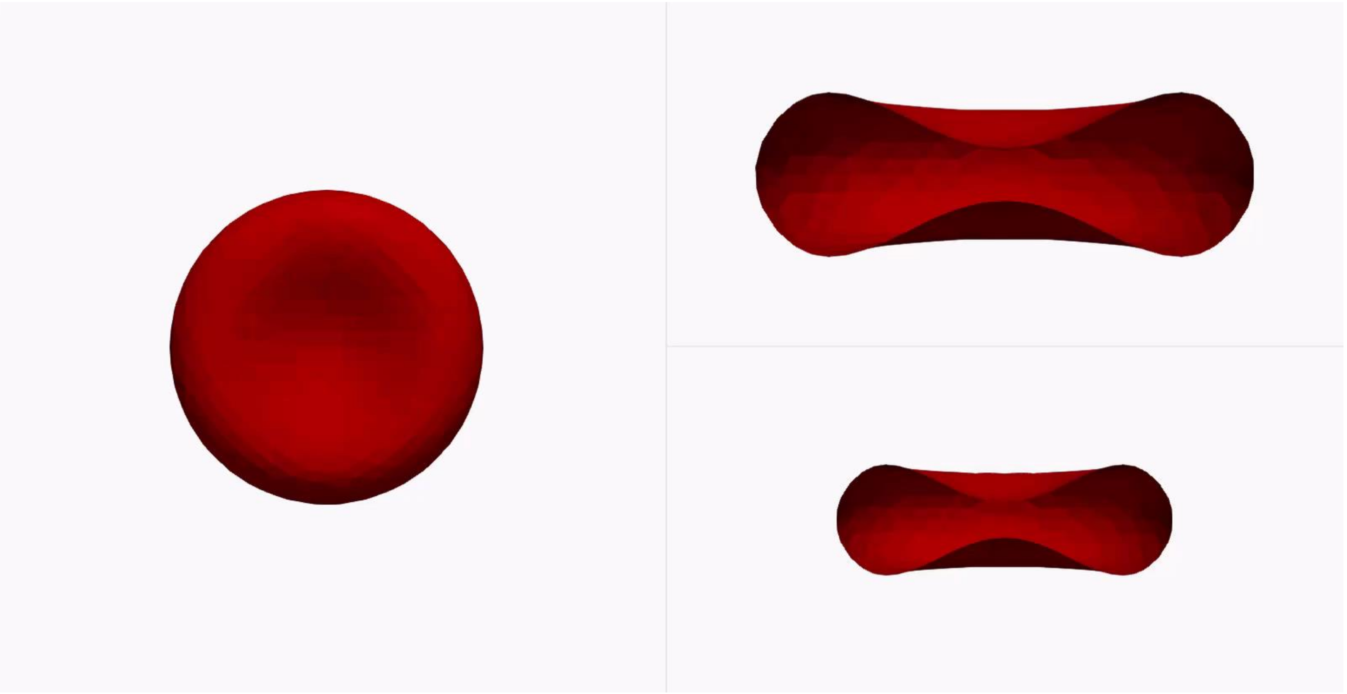




Optical tweezers experiment
by Dao, Li & Suresh

Recovery experiment: viscoelastic behavior of RBC





$t_c \sim 150\text{ms}$

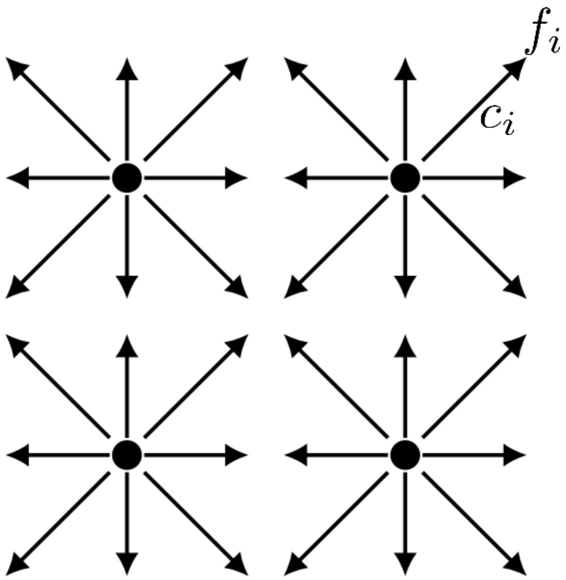
Lattice Boltzmann: Simulation of the fluid phase

$$f_i(\boldsymbol{x},t) \leftarrow f_i(\boldsymbol{x},t) + \Omega_i(\boldsymbol{x},t) \qquad (Collision)$$

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i \Delta t, t + \Delta t) = f_i(\boldsymbol{x},t) \qquad (Streaming)$$

f_i populations

c_i velocities



Regular lattice: D2Q9 | DdQq

Macroscopic properties

$$\rho(\boldsymbol{x},t) = \sum_i f_i,$$

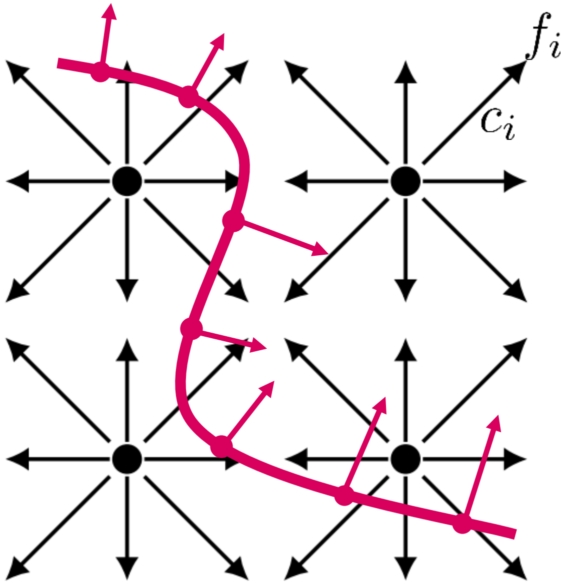
$$\rho \boldsymbol{u} = \sum_i \boldsymbol{c}_i f_i,$$

$$\boldsymbol{\sigma} = -p\mathbf{I} + \left(\frac{1}{2\tau} - 1\right) \sum_i \boldsymbol{c}_i \boldsymbol{c}_i (f_i - f_i^{eq}),$$

$$f_i^{eq} = f(\rho, \boldsymbol{u})$$

Very GPU friendly

Lattice Boltzmann: Simulation of the fluid phase



Immersed boundary

Force field, f_{imm} (IBM)

Shan-Chen forcing scheme

$$\boldsymbol{f}_{imm} \propto (\boldsymbol{U}_{npFEM} - \boldsymbol{U}_{fluid})$$

→ Force to impose the no-slip boundary condition

$$\boldsymbol{u}^G = \boldsymbol{u} + \tau \Delta t \boldsymbol{f}_{imm}$$

$$\{f_i^{eq}(\rho, \boldsymbol{u}^G)\}_{i=0}^{q-1}$$

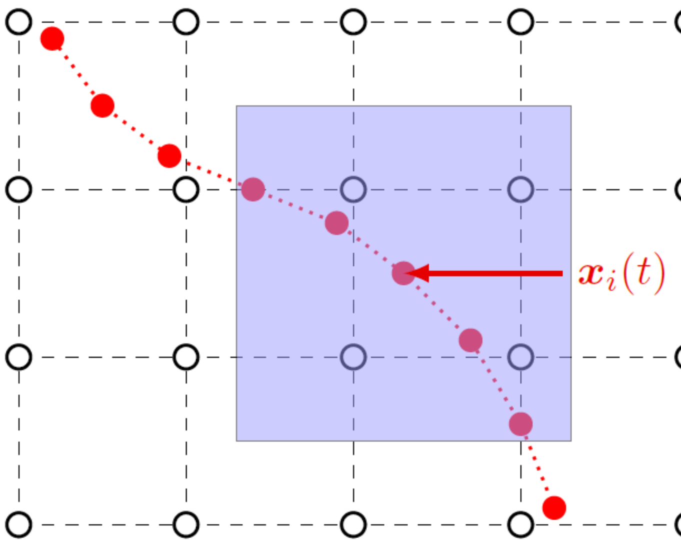
Immersed Boundary Method (IBM): Coupling the two phases

Multi-Direct Forcing Method

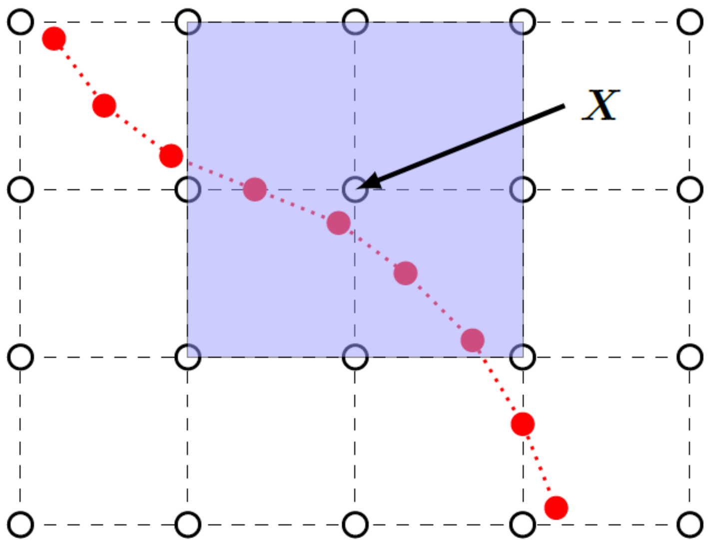
1. **Mittal R, Iaccarino G. Immersed boundary methods. Annual Review of Fluid Mechanics 2005;37(1):239–61. doi:10.1146/annurev.fluid.37.061903.175743**
2. **Wang Z, Fan J, Luo K. Combined multi-direct forcing and immersed boundary method for simulating flows with moving particles. International Journal of Multiphase Flow 2008;34(3):283–302. doi:10.1016/j.ijmultiphaseflow.2007. 10.004**
3. **Ota K, Suzuki K, Inamuro T. Lift generation by a two-dimensional symmetric flapping wing: Immersed boundary-lattice Boltzmann simulations. Fluid Dynamics Research 2012;44(4). doi:10.1088/0169-5983/44/4/045504**

Immersed Boundary Method (IBM): Coupling the two phases

Illustration of velocity interpolation (left) and force spreading (right). The velocity of vertex i is interpolated from the lattice nodes within the square region. The force acting at lattice node X is the combination of all contributions within the square region.



Not GPU friendly



$$u^*(X_k, t) = \sum_{x} u^*(x, t) W(x - X_k) \Delta x^3$$

$$f_l(X_k, t) \mathrel{+}= \frac{U_k(t) - u^*(X_k, t)}{\Delta t} A_k$$

Start

End

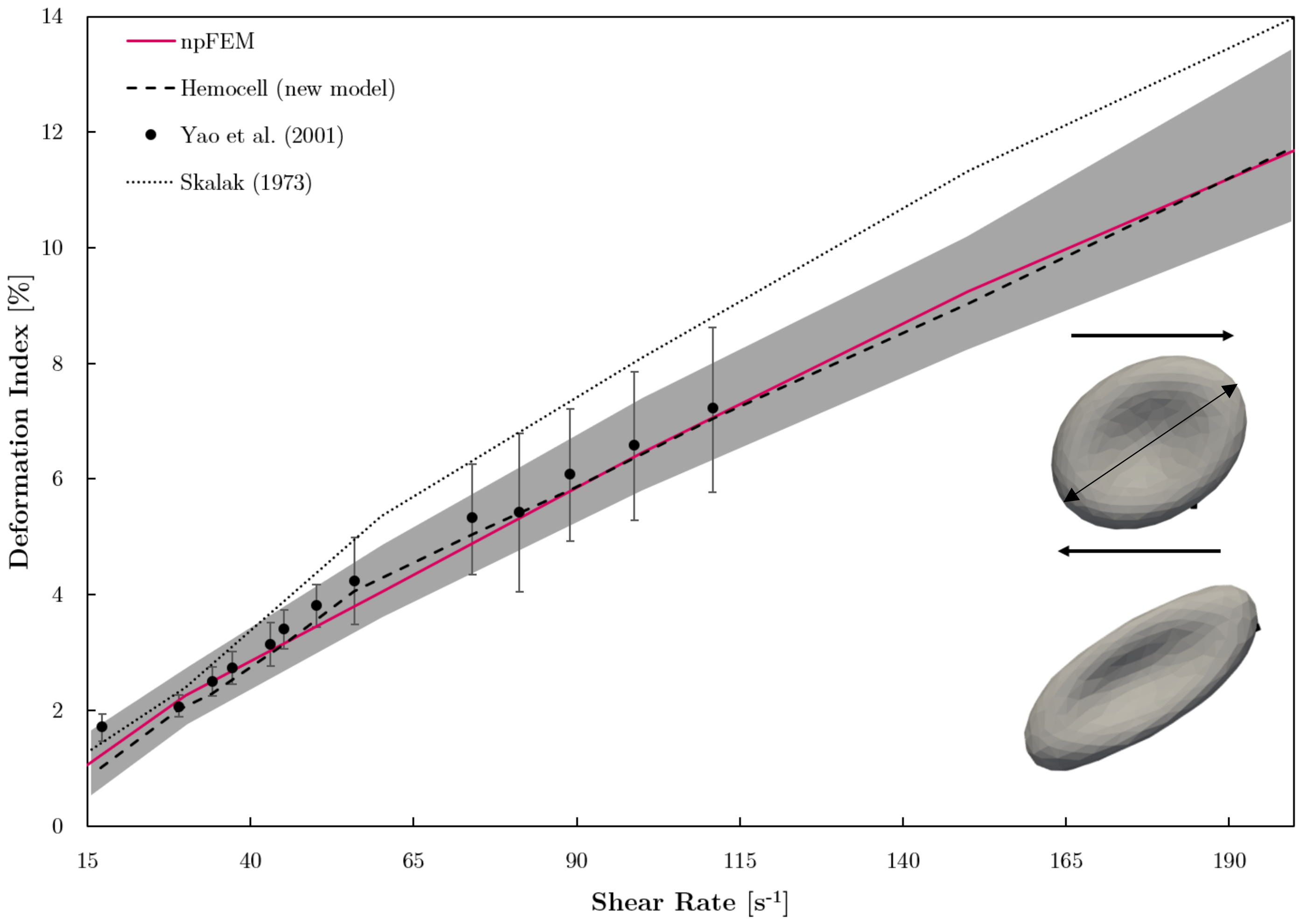
$$f_l(x, t) = \sum_{X_k} f_l(X_k, t) W(x - X_k)$$

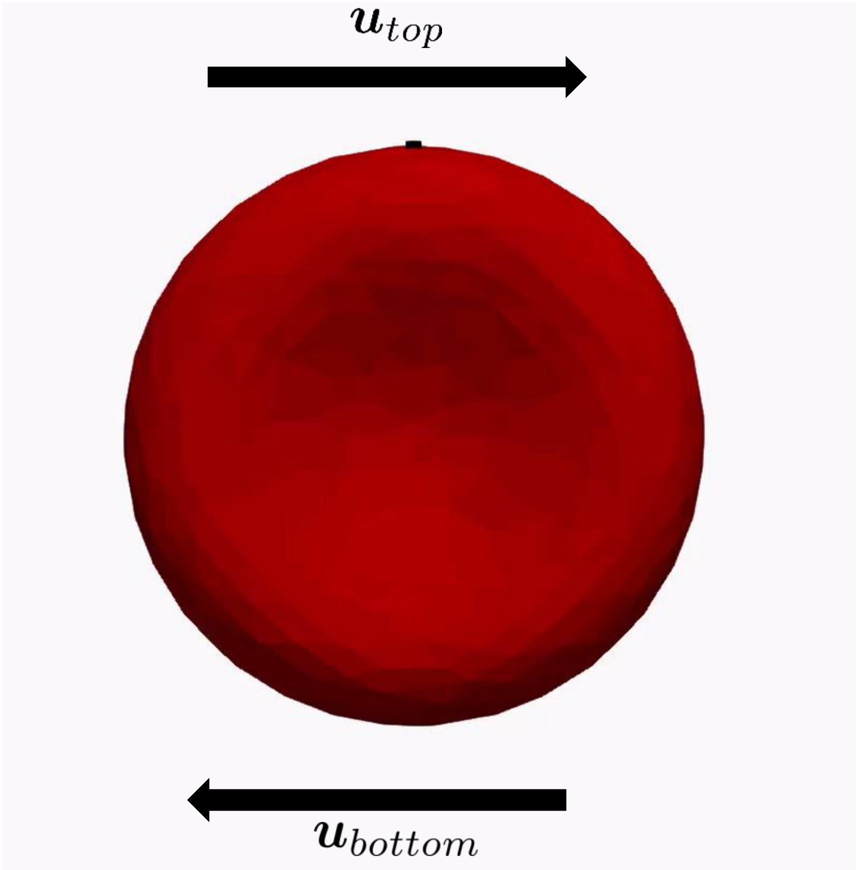
$$u_l(x, t) = u^*(x, t) + f_l(x, t) \Delta t$$

Images from Timm Krüger

Number of cycles/repetitions depends on the problem (from 1 to 4)

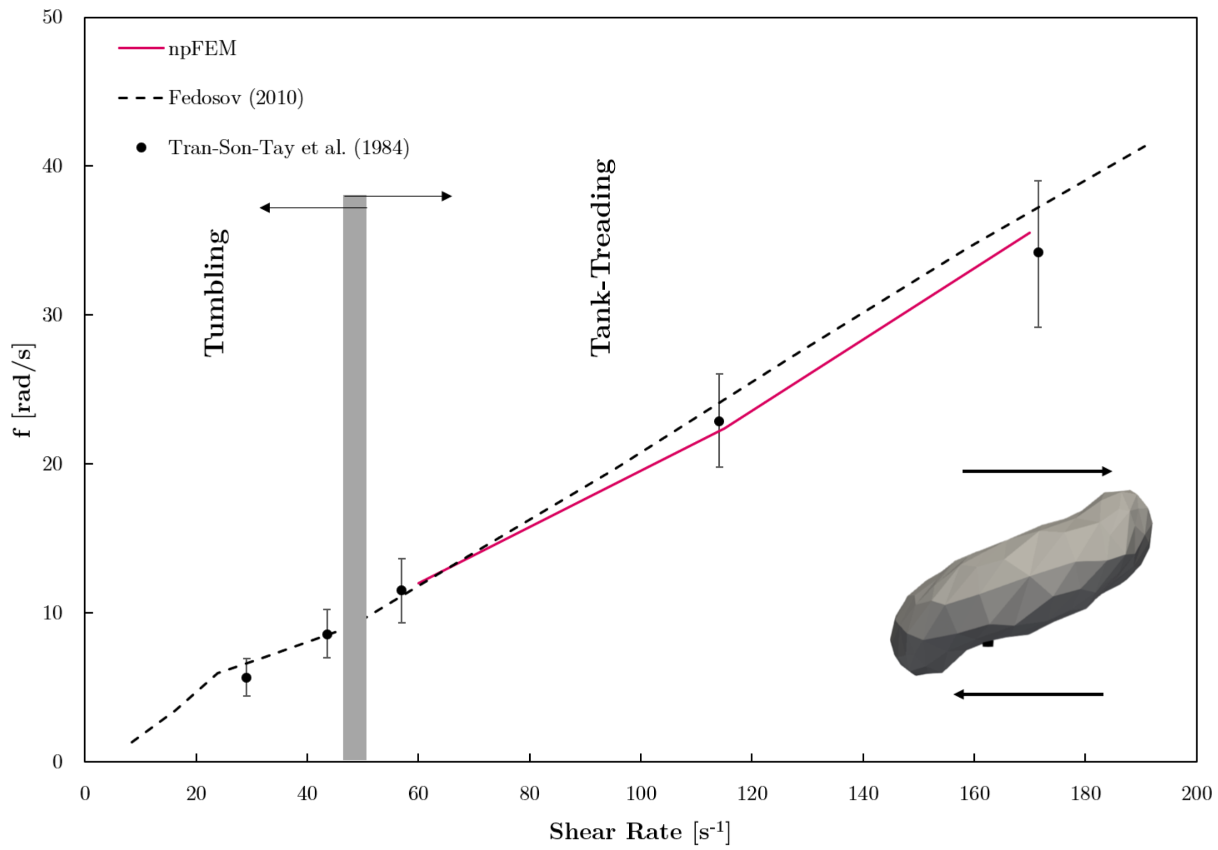
Shear flow experiment: RBC behaves like a wheel

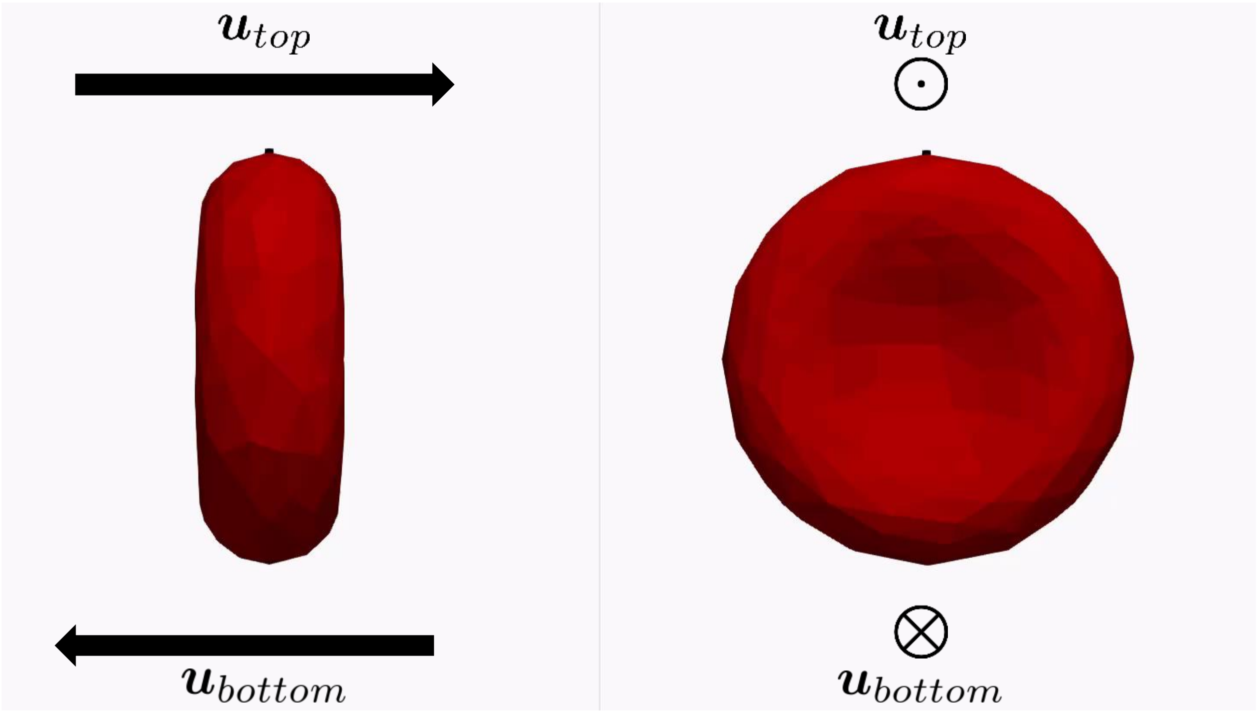




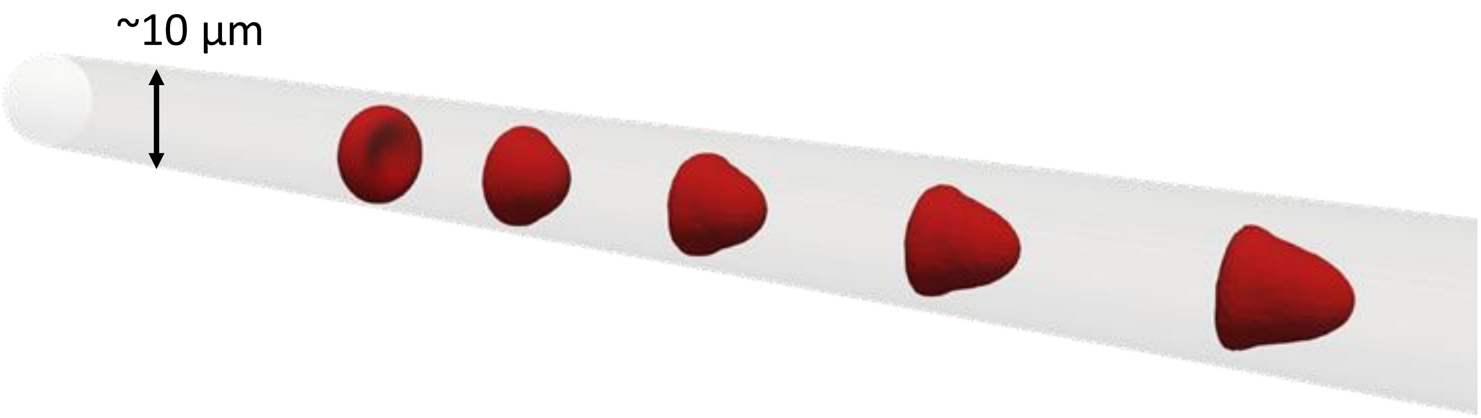
$$DI = \frac{(D_{max}/D_0)^2 - 1}{(D_{max}/D_0)^2 + 1}$$

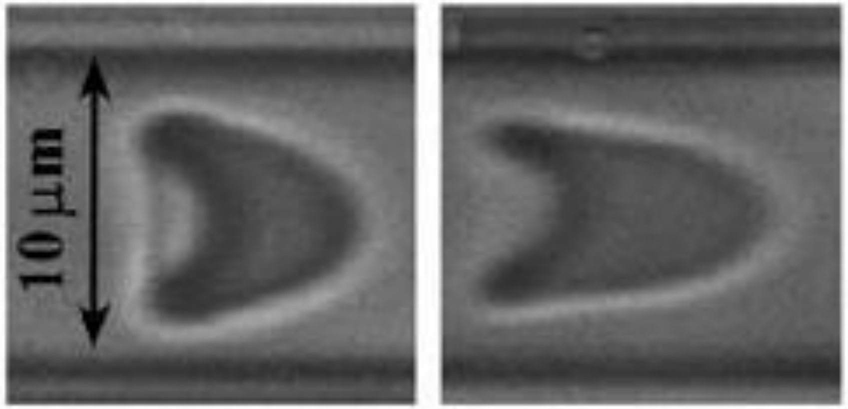
Shear flow experiment: Tank-treading



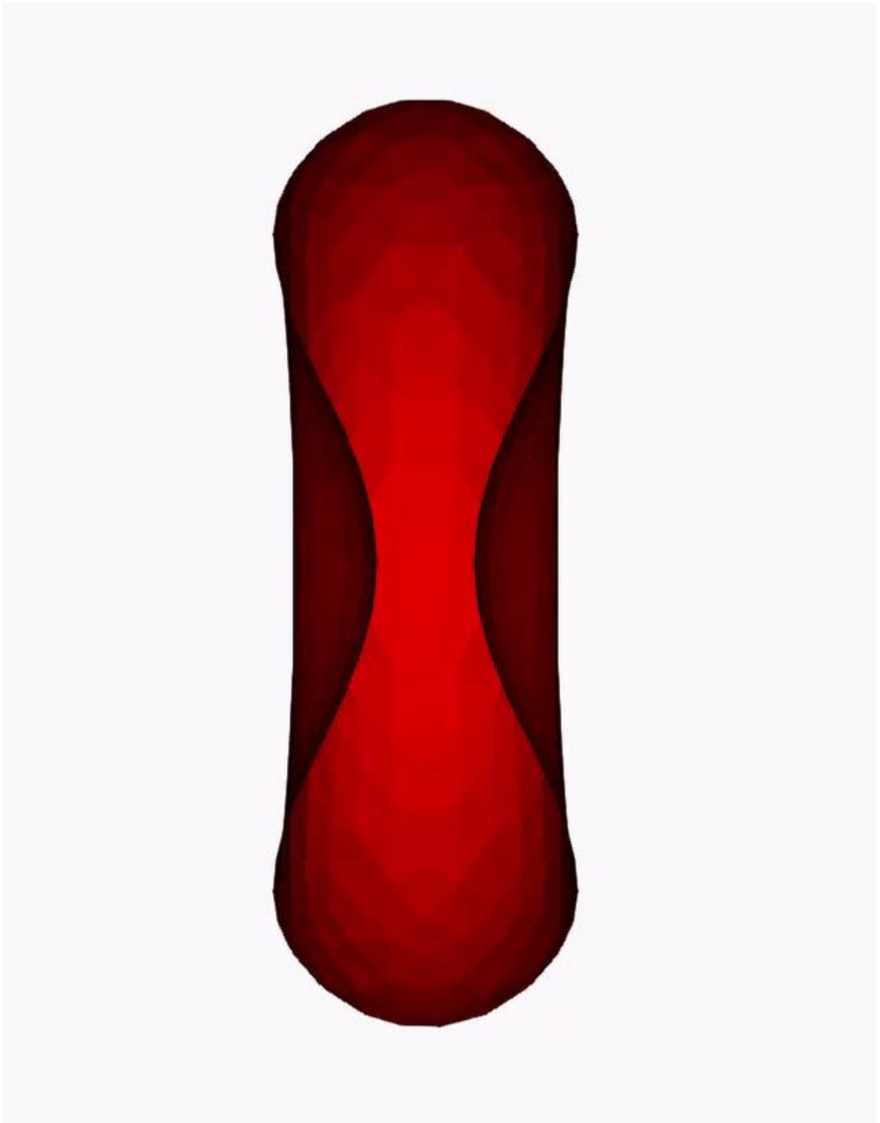


Poiseuille flow: Parachute-like shape

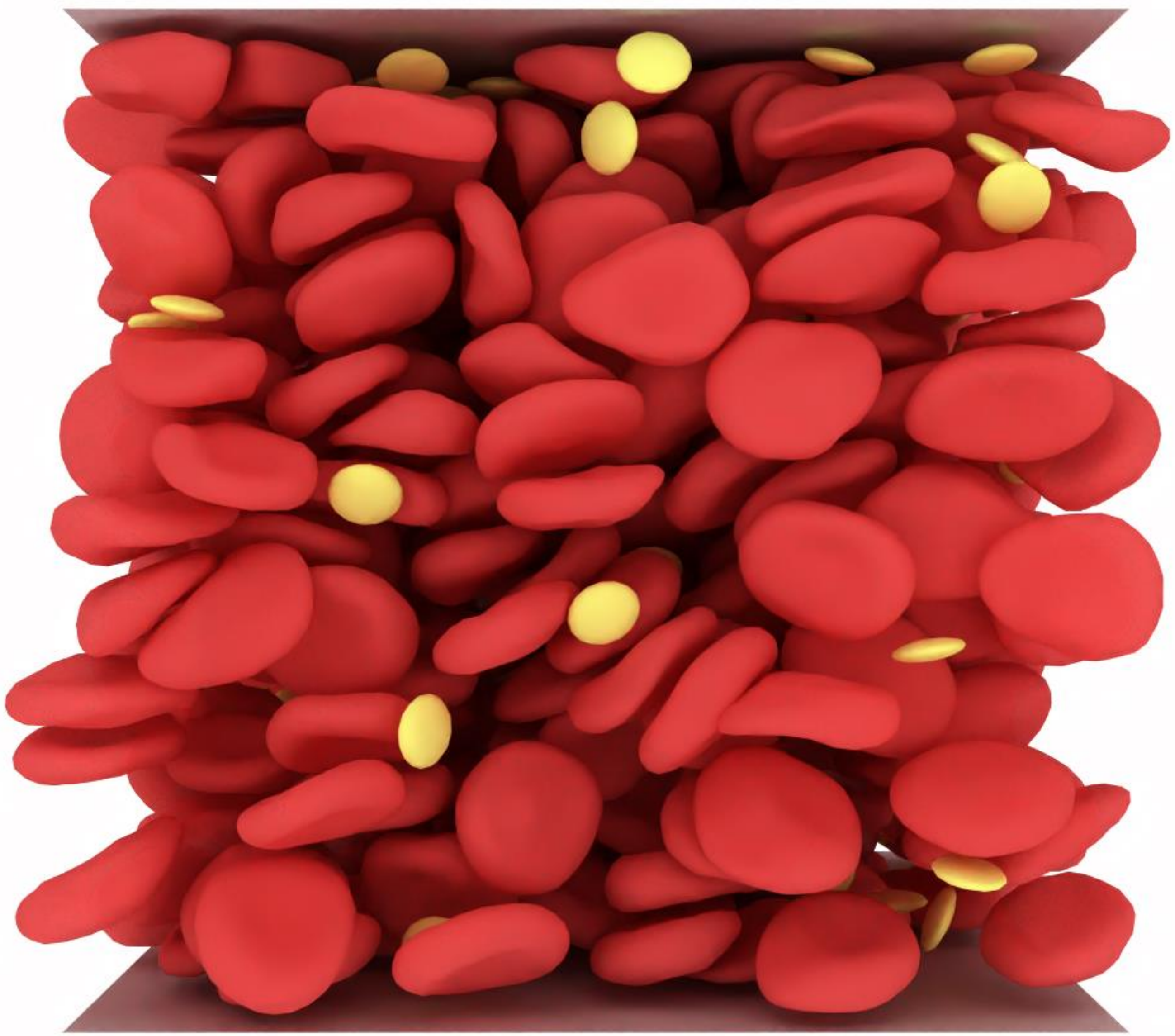




Tomaiuolo 2011



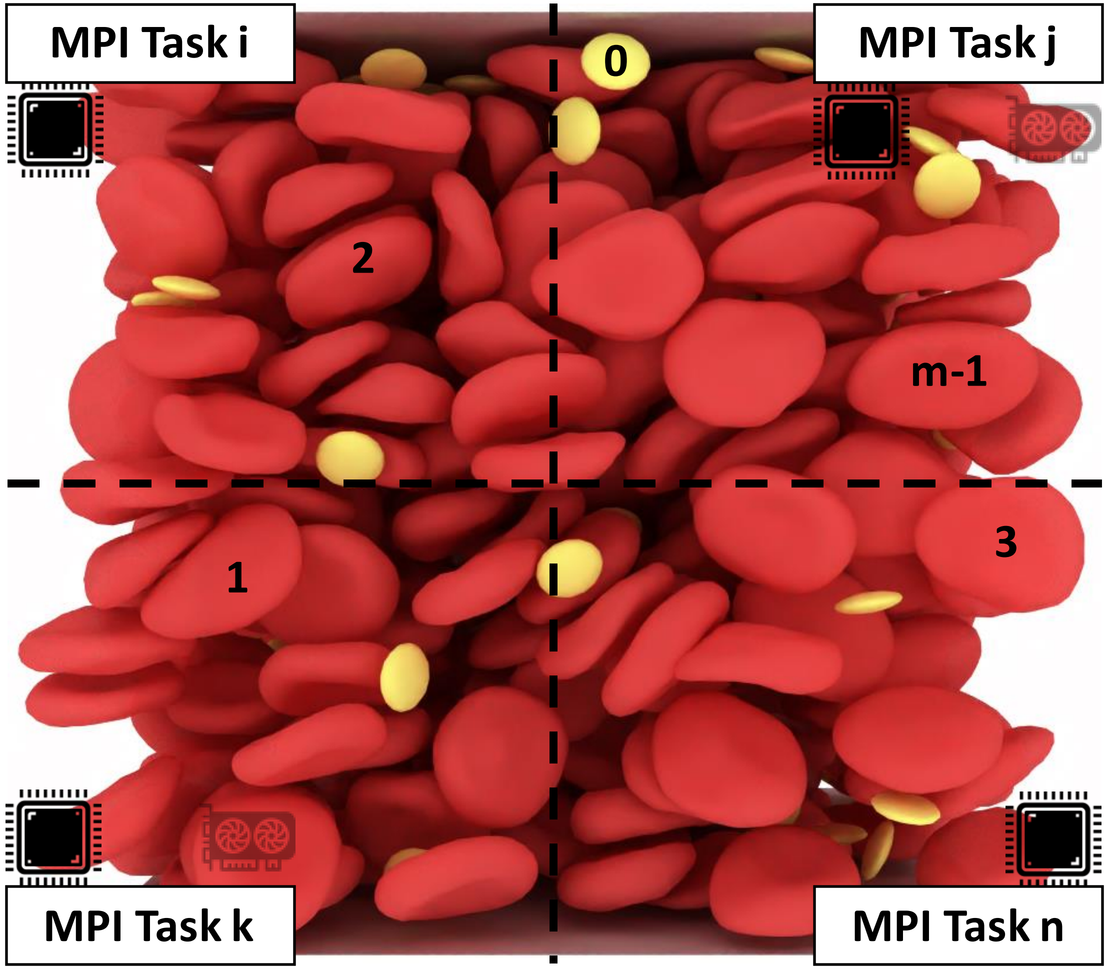
Multiple blood cells simulations



How to split the load to the available CPUs & GPUs ?

Load balancing

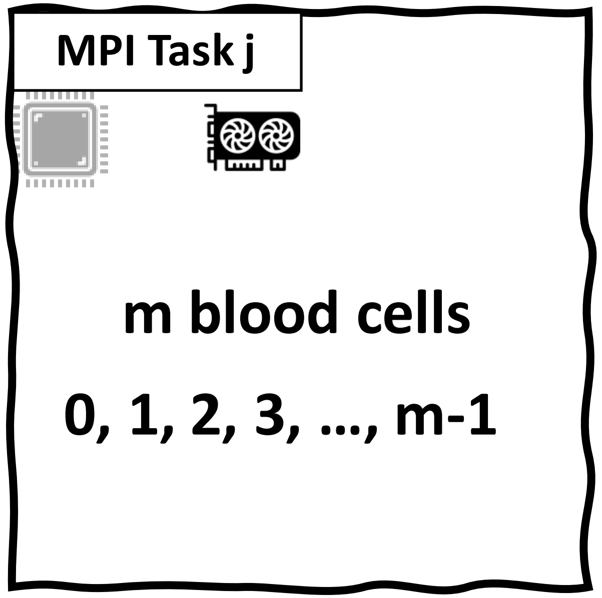
LBM &IBM



Straightforward to partition a static homogeneous grid

CPUs deal with grid points (LBM) & Lagrangian points (IBM)

npFEM



The blood cells are distributed once at the beginning to the available GPUs

They can be spatially everywhere

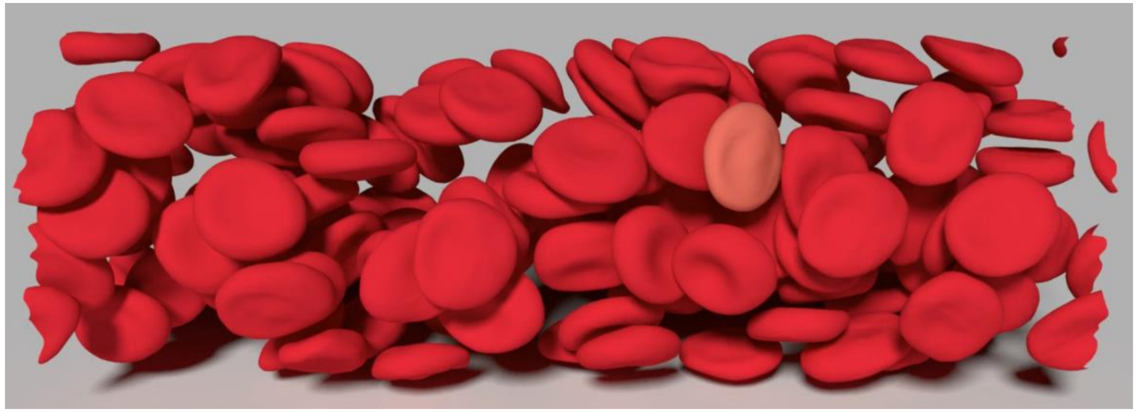
MPI point-to-point communication

- The fluid solver sends forces & collision data to the solid solver
- The solid solver communicates the state at t+1

If no GPU-support, then the npFEM solvers are distributed to the available MPI tasks

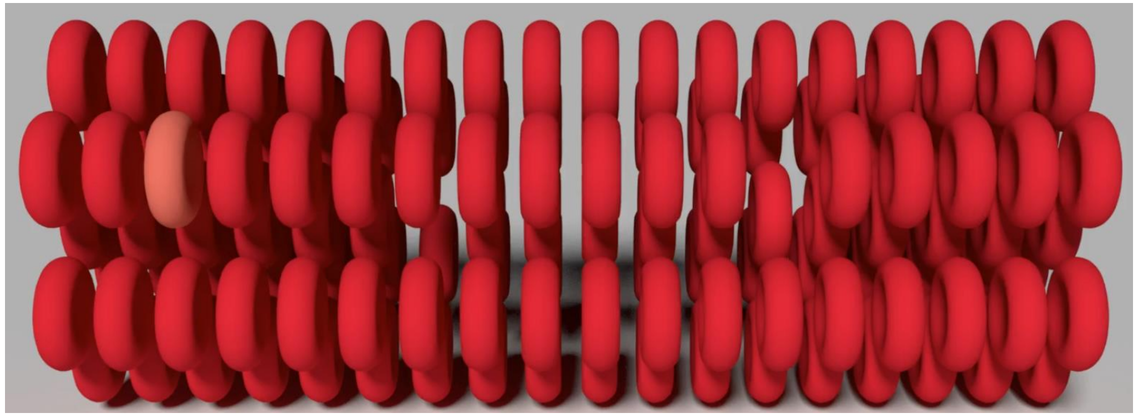
Load balancing

CPUs



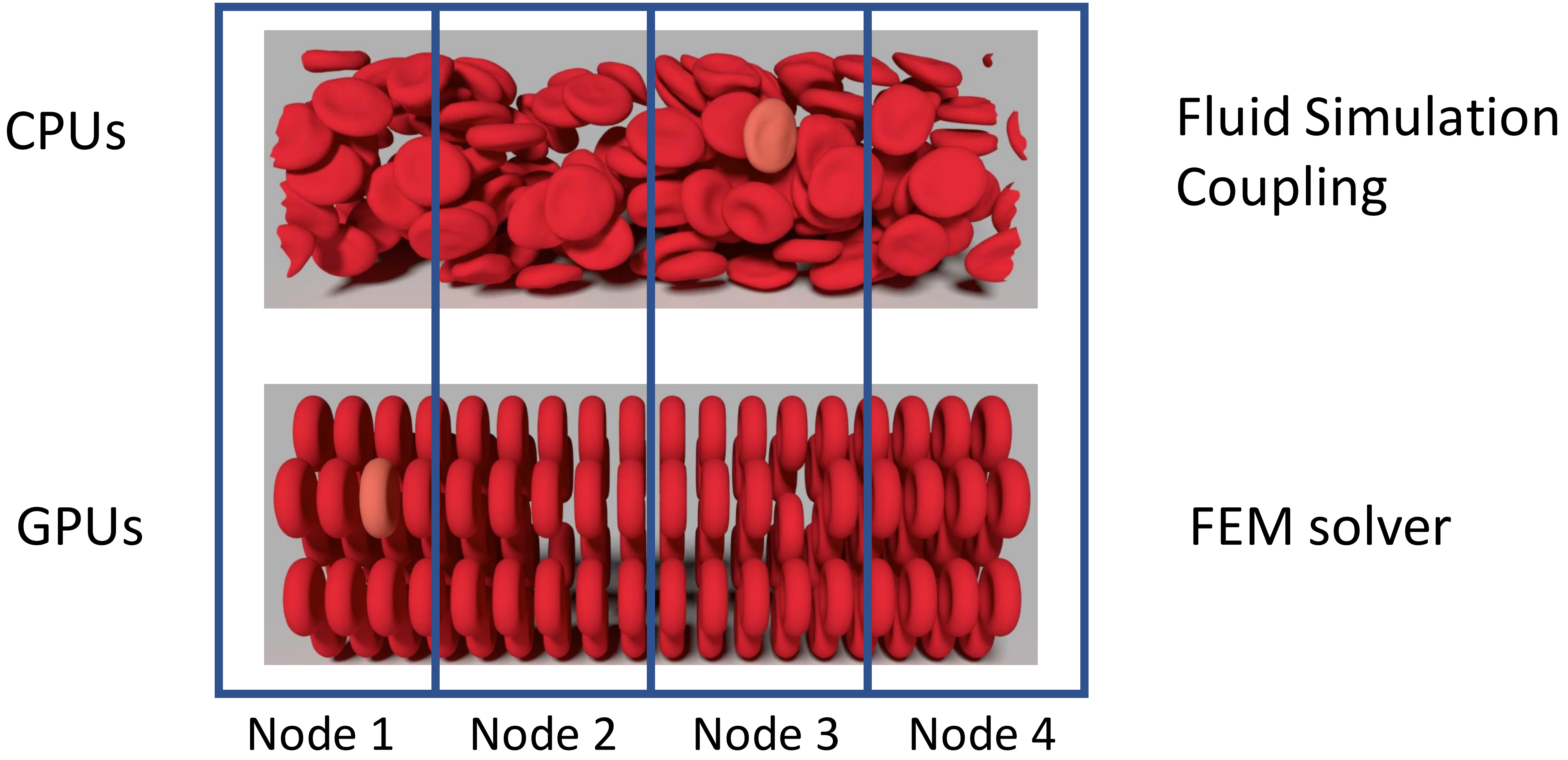
Fluid Simulation
Coupling

GPUs

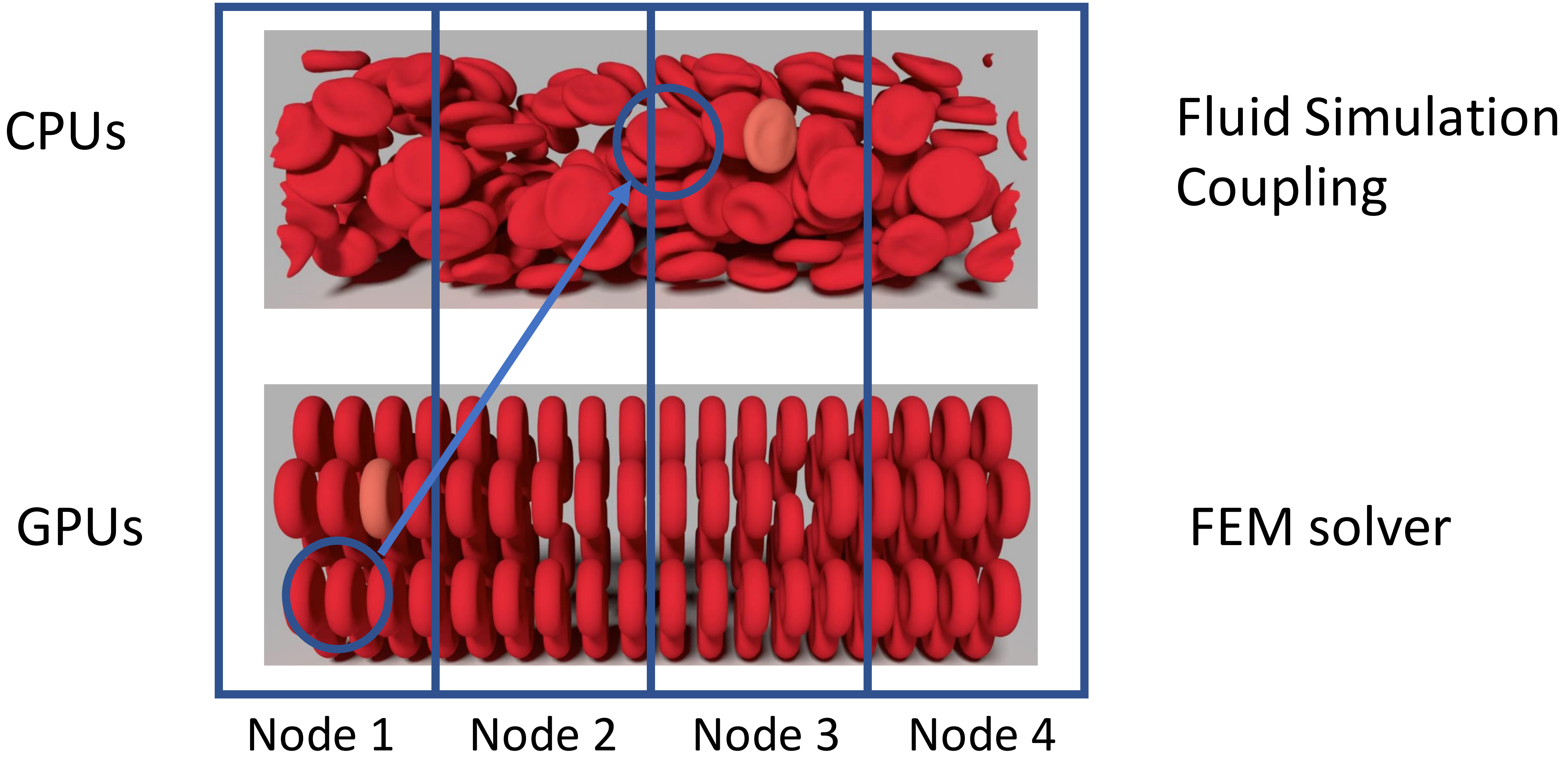


FEM solver

Load balancing



Load balancing



Generic Character of the framework

The Framework
(Thin Layer)

Fluid Solver
any type:
LBM, DPD, SPH

Solid Solver
any type:
FEM, MSS, DEM

Communication between solvers taken care by the framework
Plug-and-Play

The same principles can be applied in any FSI application

GPU acceleration

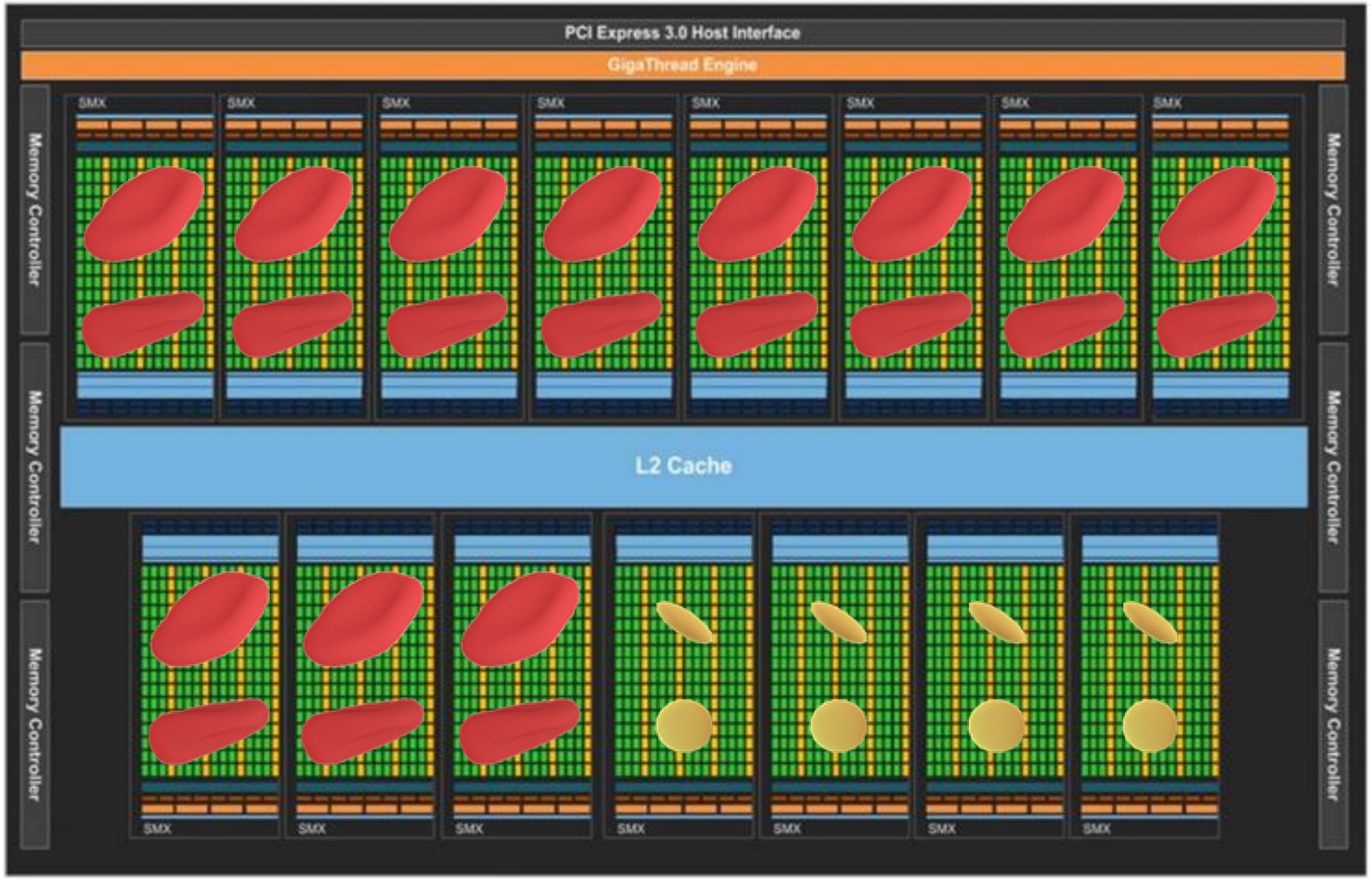
npFEM

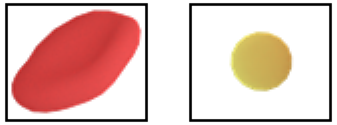
MPI Task j

m blood cells

0, 1, 2, 3, …, m-1

Fit bodies in the Streaming Multiprocessors to gain from the fast shared-memory and thread synchronization

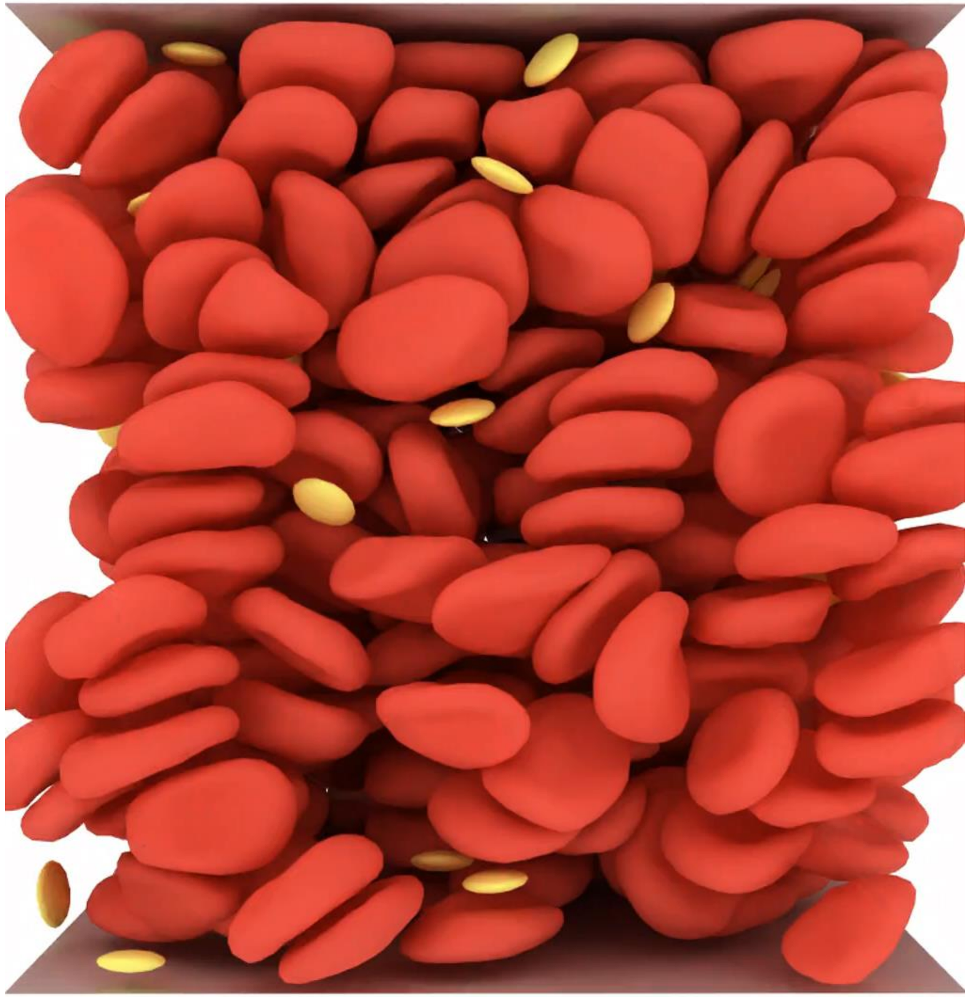




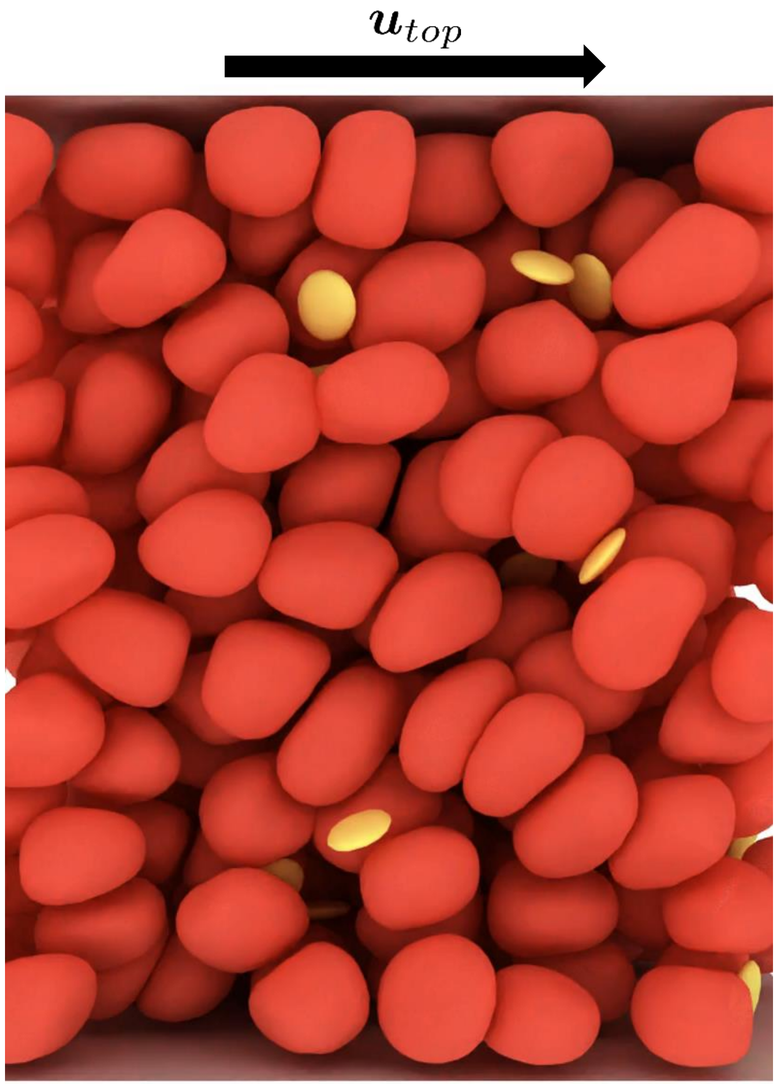
1 CUDA Block per blood cell

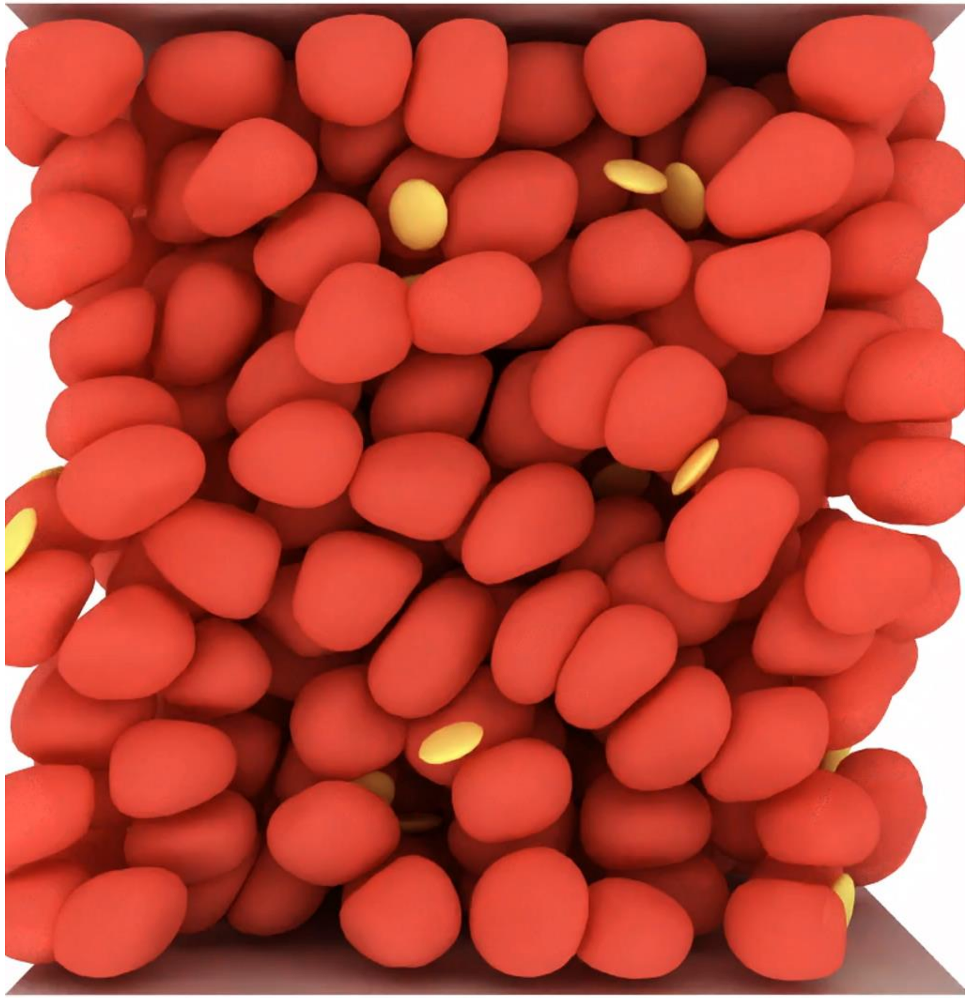
1 SMX deals with 1 CUDA block per time

Shear flow: >500 blood cells | Hematocrit 35% | Box 50x50x50 μm^3



Focus on Pathological conditions, like Diabetes (swollen RBCs)





Performance measures

Piz Daint @CSCS

No. 6 Worldwide

No. 1 Europe

GPU Node (5704 Nodes)

Hybrid Node (CPU+GPU)

CRAY XC50

12 cores – 64GB RAM

(Intel Xeon E5-2690 v3 @ 2.60GHz)

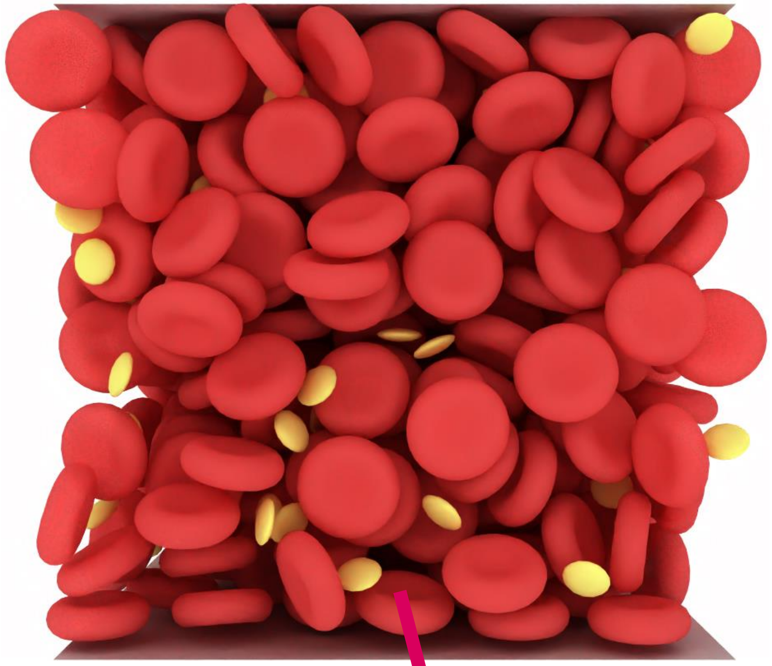
+

1 NVIDIA Tesla P100 16GB

Weak Scaling

Reference case study

Ht 35%, Box 50x50x50 μm^3



RBCs 258 surface vertices
PLTs 66 surface vertices

box50x50x50 : **1**
on 5 GPUs
RBCs: 476
PLTs: 95

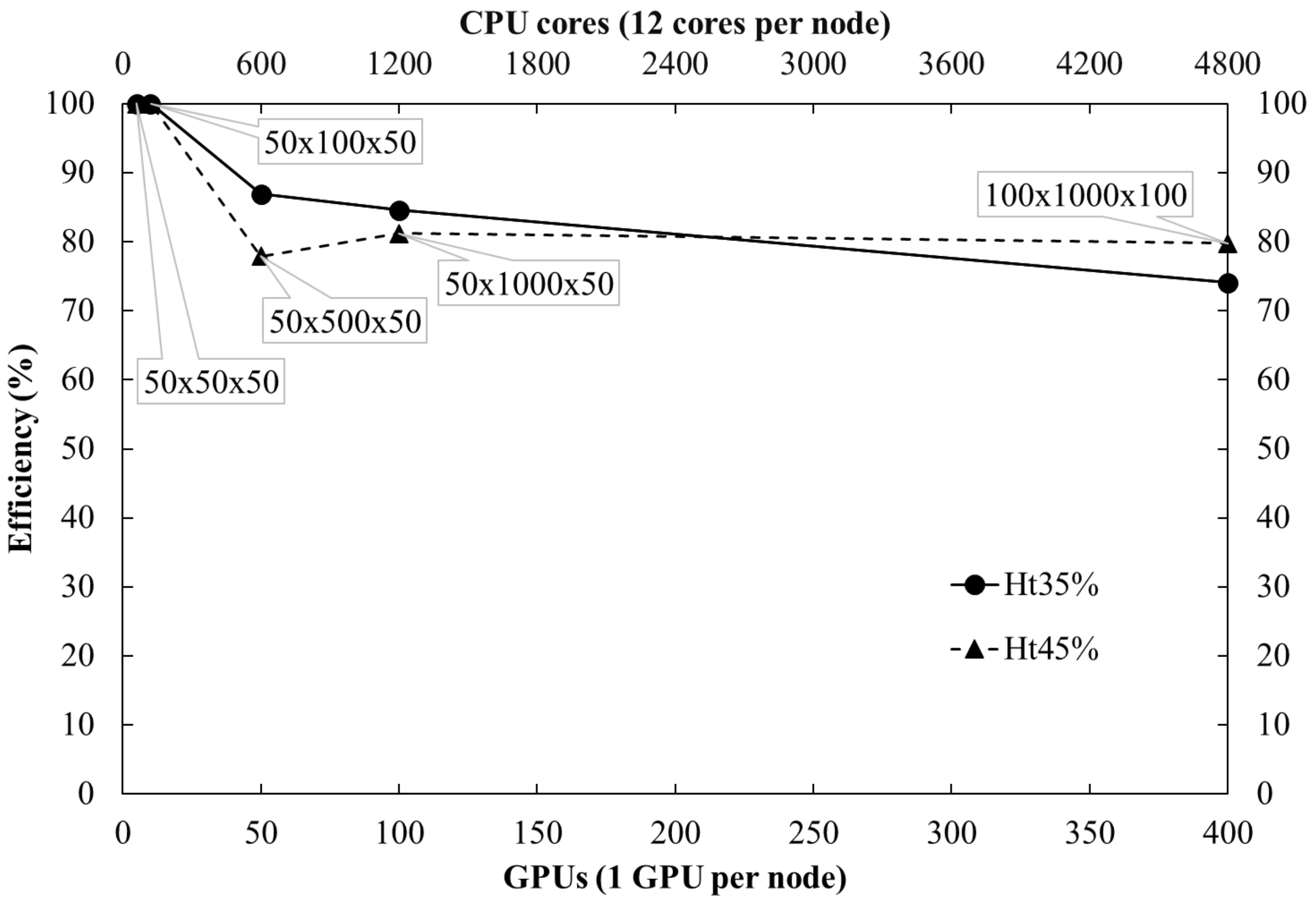
box50x100x50 : **2**
on 10 GPUs
RBCs: 953
PLTs: 190

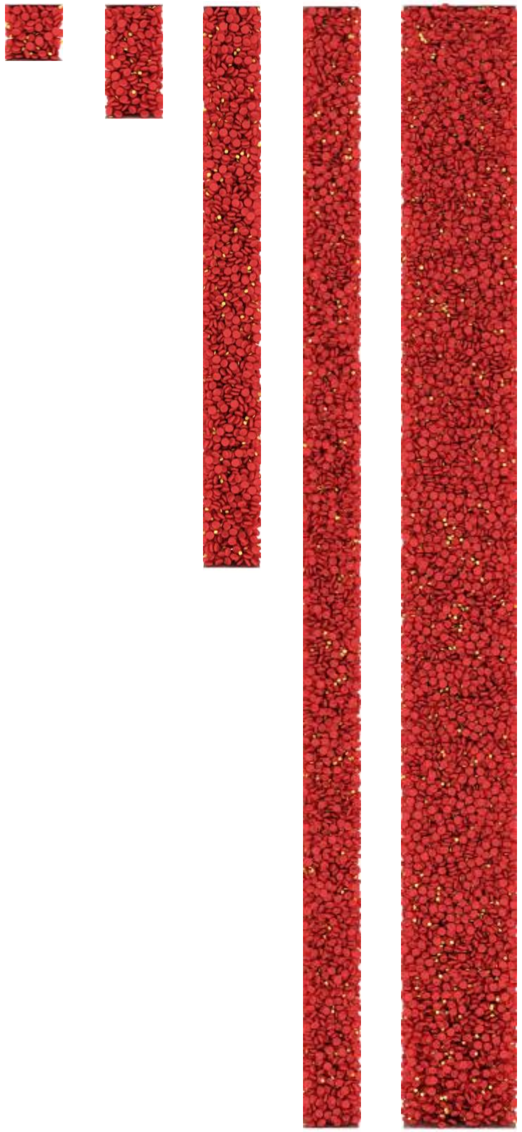
box50x500x50 : **10**
on 50 GPUs
RBCs: 4765
PLTs: 953

box50x1000x50 : **20**
on 100 GPUs
RBCs: 9531
PLTs: 1906

box100x1000x100 : **80**
on 400 GPUs
RBCs: 38126
PLTs: 7625

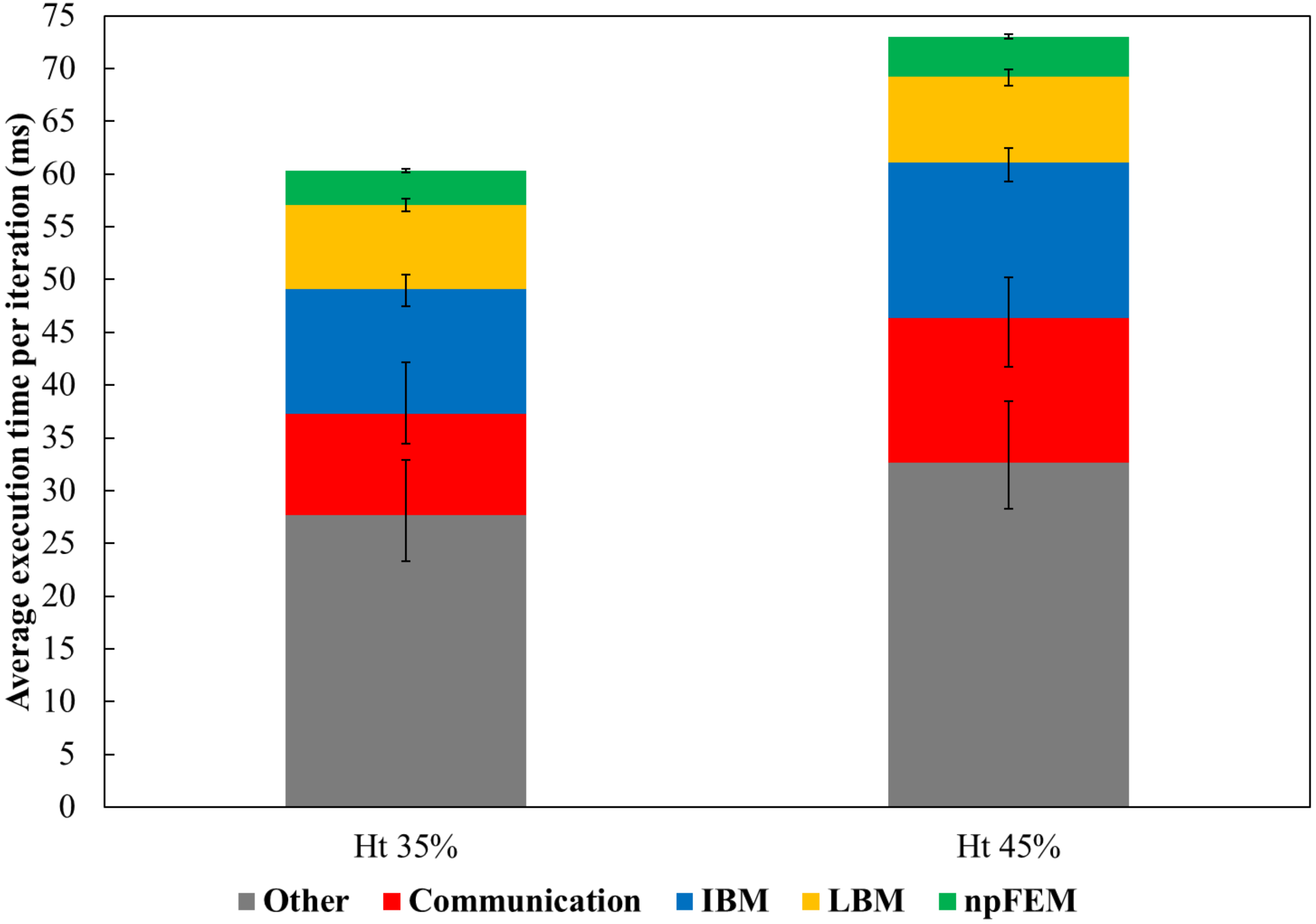
Weak Scaling – Hybrid Version



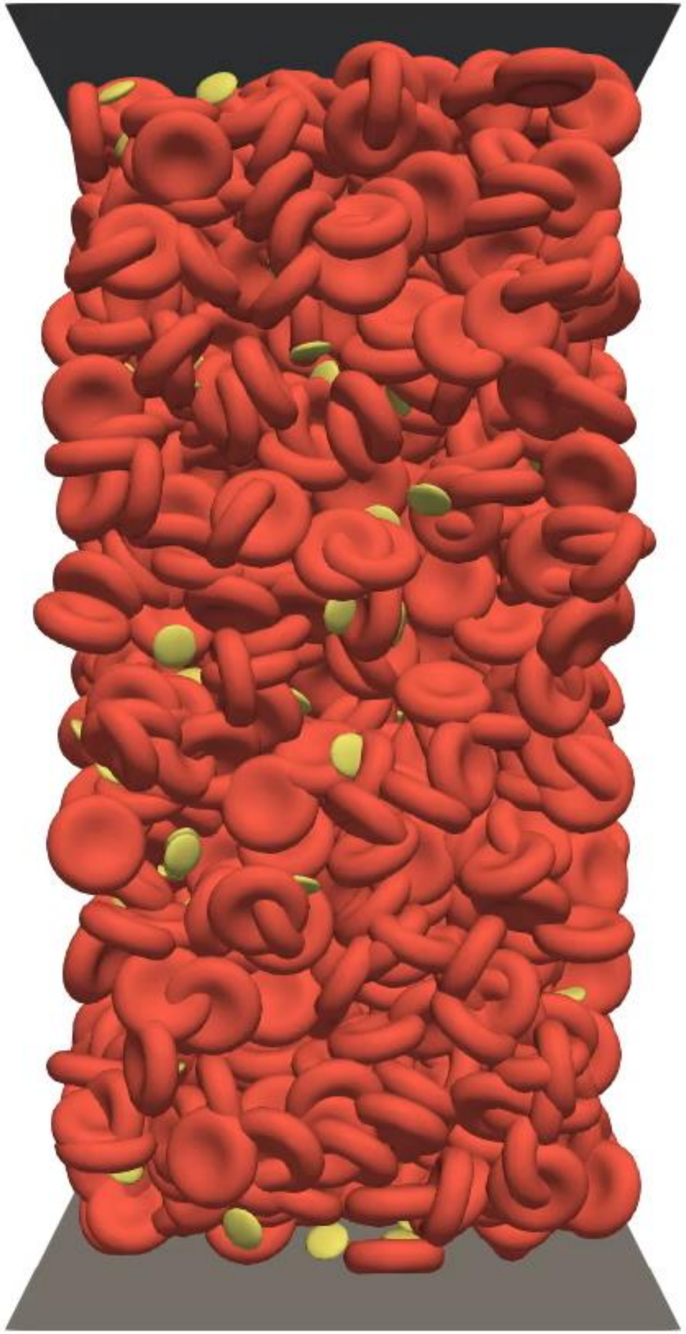


$$\text{Efficiency} = \frac{t_{N_0}}{t_N}$$

Weak Scaling – Performance of various modules



Cell Packing



Execution order of the different modules

1. Compute the macroscopic fluid properties: density, momentum, stress tensor
2. Use the stress tensor to compute the external forces on the solids from the fluid (at t)
3. Apply the immersed boundary method (bodies at t)
4. Impose the IBM force and any other forcing term to the fluid through the Shan-Chen forcing scheme
5. Collide & Stream, lattice Boltzmann steps, advance the fluid from t to t+1
6. Use the external force (step 2), solve the immersed bodies, thus advance the solids from t to t+1

Execution order of the different modules

Palabos actions class (container of operations):

1. Actions3D action#
2. Register involved Blocks (e.g., lattice, rho, j, Particles)
3. Add Data Processor in the action
4. Communication between atomic blocks (if needed)
5. action#.execute()

Check the **bloodFlowDefoBodies.cpp main time loop**:

- actions1: BoxRhoBarJPiNeqfunctional3D
- actions2a: ConstructLocalMeshesFromParticles
- actions2b: CollisionsForcesCombo
- actionsForcingTerm: AddConstForceToMomentum3D (Poiseuille)
- actions3: MultiDirectForcingImmersedBoundaryIteration3D
- actions4: ExternalRhoJcollideAndStream3D
- actions5: LocalMeshToParticleVelocity3D
- actions6: AdvanceParticlesEveryWhereFunctional3D

UNIVERSITÉ DE GENÈVE

Questions?

PALABOS