A shared memory parallel implementation of the Eulerian-Lagrangian coupling equations for a Stokes Solver

Sachin Natesh

Courant Institute, NYU

July 20, 2020

Stokes flow and Eulerian-Lagrangian coupling

- Suppose we have N_p particles with *Lagrangian* coordinates $\boldsymbol{y}_k$ immersed in a Stokes fluid in a triply periodic domain Ω. Consider a uniform grid G on Ω with spacing h. The N^3 grid nodes $\boldsymbol{x} \in G$ are called *Eulerian* coordinates.
- Given forces $\boldsymbol{f}(\boldsymbol{y}_k)$ on the particles, what is their velocity?

$$\eta \nabla^2 \boldsymbol{u}(\boldsymbol{x}) - \nabla p(\boldsymbol{x}) = -\boldsymbol{f}(\boldsymbol{x}), \quad \nabla \cdot \boldsymbol{u}(\boldsymbol{x}) = 0. \tag{1}$$

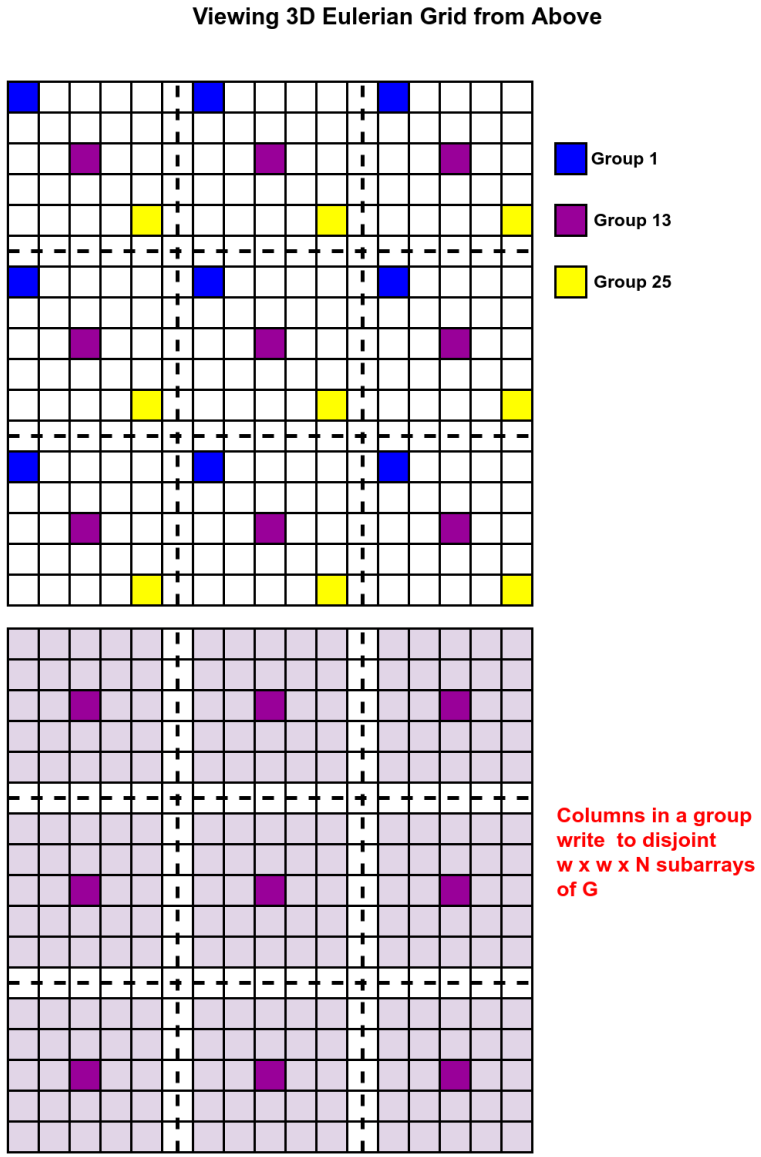
- We need a way to *spread* forces on the particles to the Eulerian grid, solve (1), and *interpolate* the fluid velocity $\boldsymbol{u}(\boldsymbol{x})$ locally around each particle:

$$\textbf{spread } \forall \boldsymbol{x} \in G: \quad \boldsymbol{f}(\boldsymbol{x}) = \sum_{k=1}^{N_p} \boldsymbol{f}(\boldsymbol{y}_k) \Delta(\boldsymbol{x} - \boldsymbol{y}_k) \tag{2}$$

$$\textbf{interpolate } \forall k: \quad \boldsymbol{v}(\boldsymbol{y}_k) = \sum_{\boldsymbol{x} \in G} \boldsymbol{u}(\boldsymbol{x}) \Delta(\boldsymbol{x} - \boldsymbol{y}_k) h^3 \tag{3}$$

Domain decomposition

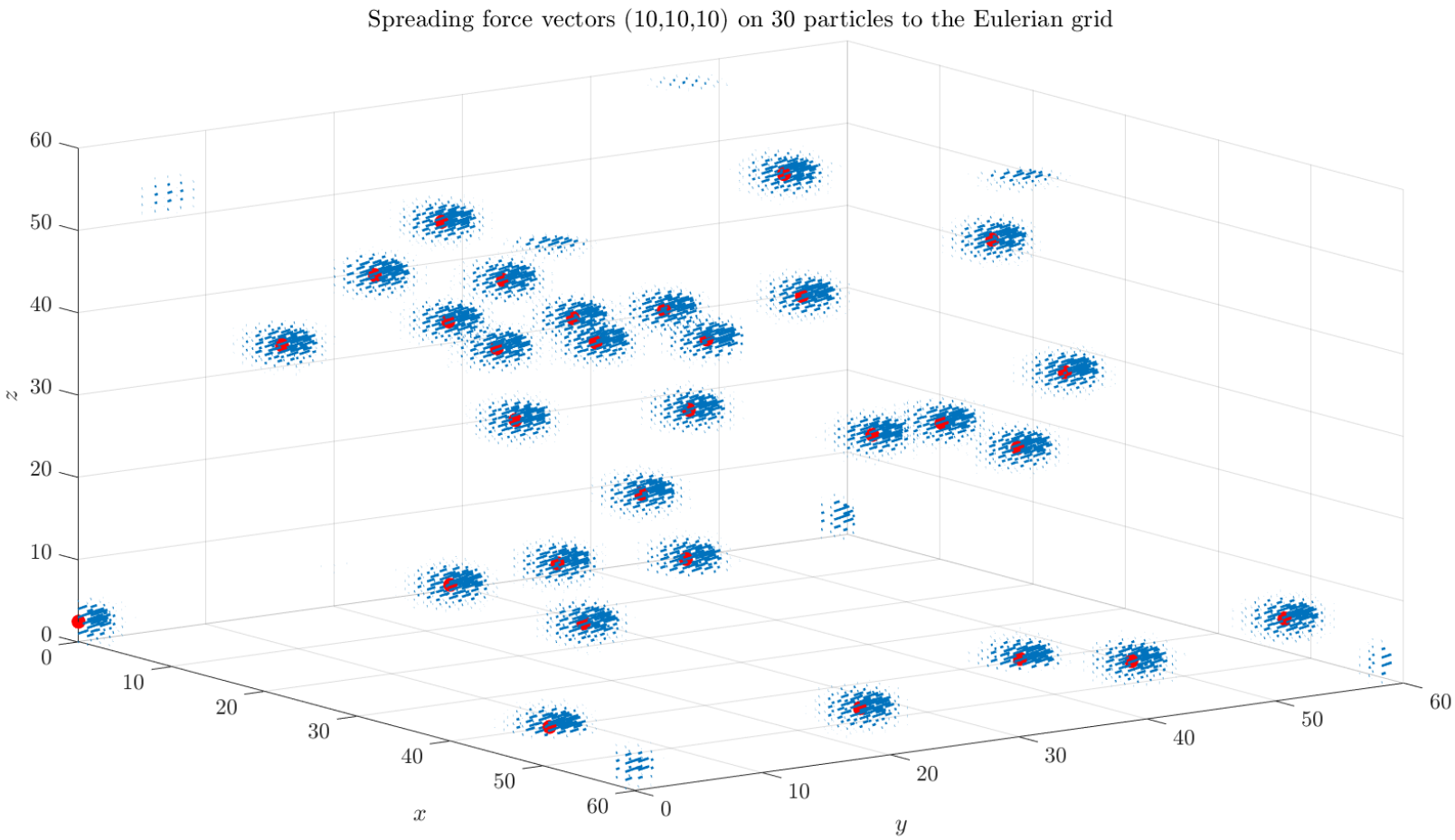
- In (2)-(3), Δ is a *finitely supported, symmetric spreading 'kernel'* of width wh, for $w \in \mathbb{Z}^+$, i.e. if $|x^i - y_k^i| > wh/2$ for $i = 1, 2$ or 3, then $\Delta(\boldsymbol{x} - \boldsymbol{y}_k) = 0$.
- This means a Lagrangian coordinate $\boldsymbol{y}_k$ only 'talks' to a certain $w \times w \times w$ subarray of the Eulerian grid G.
- All particles above and below $\boldsymbol{y}_k$ and within $h/2$ in x, y talk to the *same* $w \times w \times N$ subarray of $G \Rightarrow$ partition the 3D grid into groups of columns!
- Loop over the w^2 groups of columns and parallelize over the columns in each group.



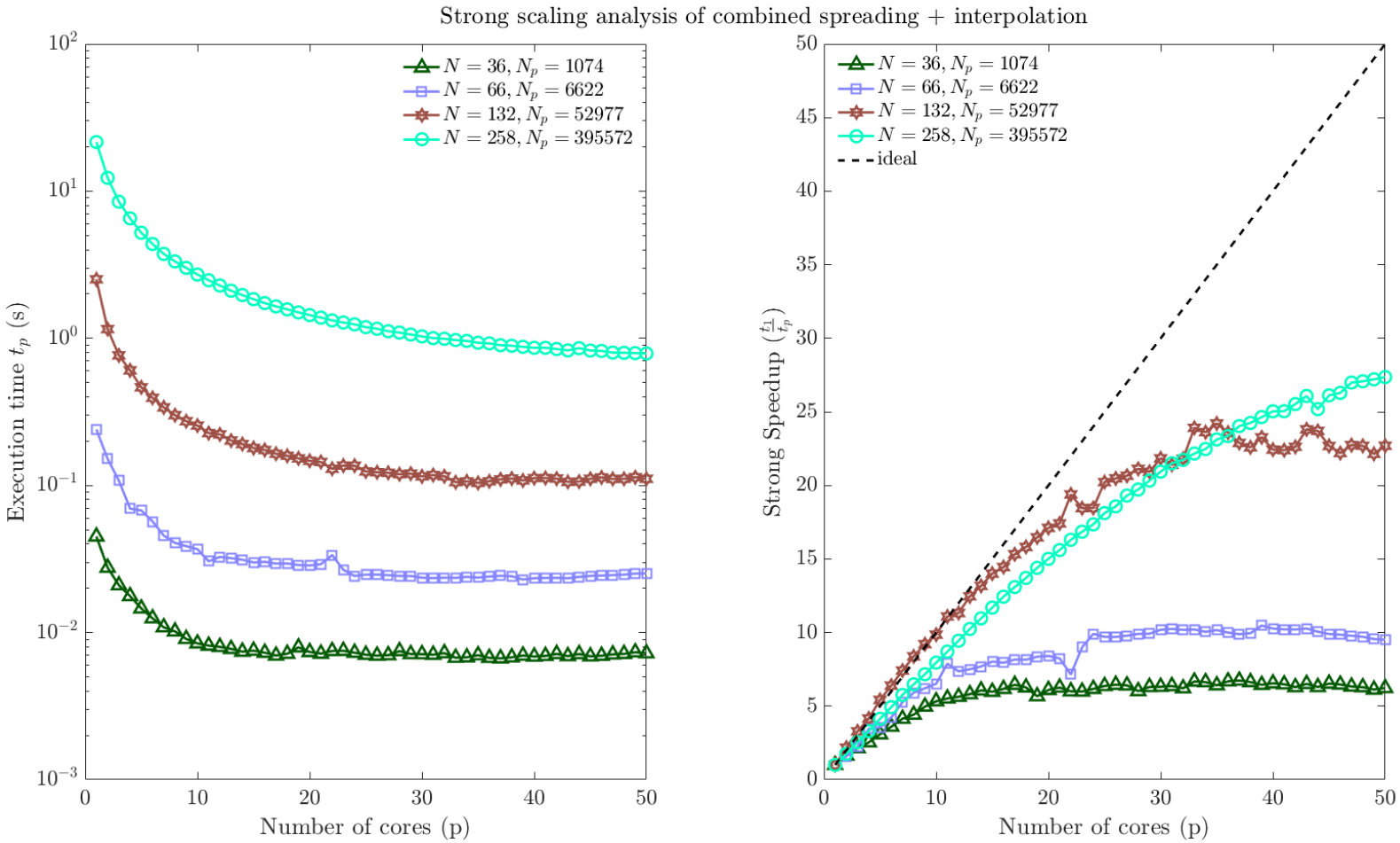
Algorithm outline

1. Construct an array `first(i,j)` that gives the *index k* of the first particle in column i, j. Along with this, construct an array `next(k)` that gives the *index* of the *next particle in the column with particle k*.
2. If there is a particle in the column, `gather` the lagrangian coordinates and forces from the global arrays into cache-aligned local ones.
3. Compute the global indices of the $w \times w \times N$ subarray of G influenced by the column. Use them to `gather` the Eulerian forces for one column into cache-aligned memory.
4. Get the $w \times w \times w$ kernel weights for each particle in the column $\Rightarrow$ vectorize over particles with `#pragma omp simd`.
5. Use the kernel weights and Lagrangian forces to update the Eulerian forces for the column $\Rightarrow$ vectorize over Eulerian points.
6. `scatter` the results back the global Eulerian grid.
7. Interpolation is just the weighted adjoint of spreading, so the implementations mirror each other somewhat.

Results


Spreading force vectors (10,10,10) on 30 particles to the Eulerian grid
z
x
y
0
10
20
30
40
50
60

Results


Strong scaling analysis of combined spreading + interpolation
Execution time t_p (s)
Number of cores (p)
$N = 36, N_p = 1074$
$N = 66, N_p = 6622$
$N = 132, N_p = 52977$
$N = 258, N_p = 395572$
Strong Speedup ($\frac{t_1}{t_p}$)
ideal

References

[MP97]

David M. Mcqueen and Charles S. Peskin, *Shared-memory parallel vector implementation of the immersed boundary method for the computation of blood flow in the beating mammalian heart*, Journal of Supercomputing **11** (1997), no. 3, 213–236 (English (US)).