



INTERNATIONAL SCIENTIFIC CONFERENCE «RUSSIAN SUPERCOMPUTING DAYS 2026»

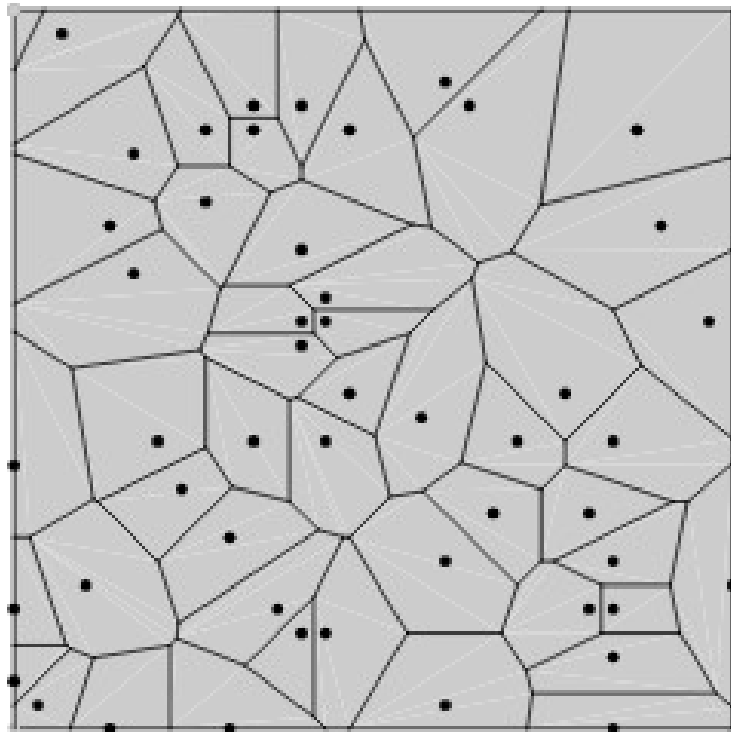
PARALLEL COMPUTATION OF DENSITY MAPS IN MOLECULAR DYNAMICS BASED ON VORONOI TESSELLATION

VASILY PISAREV

RUFINO HAROLDO LOCON AMEZQUITA

INTERNATIONAL LABORATORY FOR SUPERCOMPUTER ATOMISTIC
MODELING AND MULTI-SCALE ANALYSIS (HSE-MIEM)

September 28th 2026



Voronoi Tessellation — a geometric construction that splits the space or region of space into cells, for a set of point particles. One of the main constructions is the Cell, which is the space closer to a given particle than to any other.

Some existing software implementations (in 3D):

- Qhull
- CGAL
- Voropp
- TetGen



DOMAIN DECOMPOSITION

Assumptions

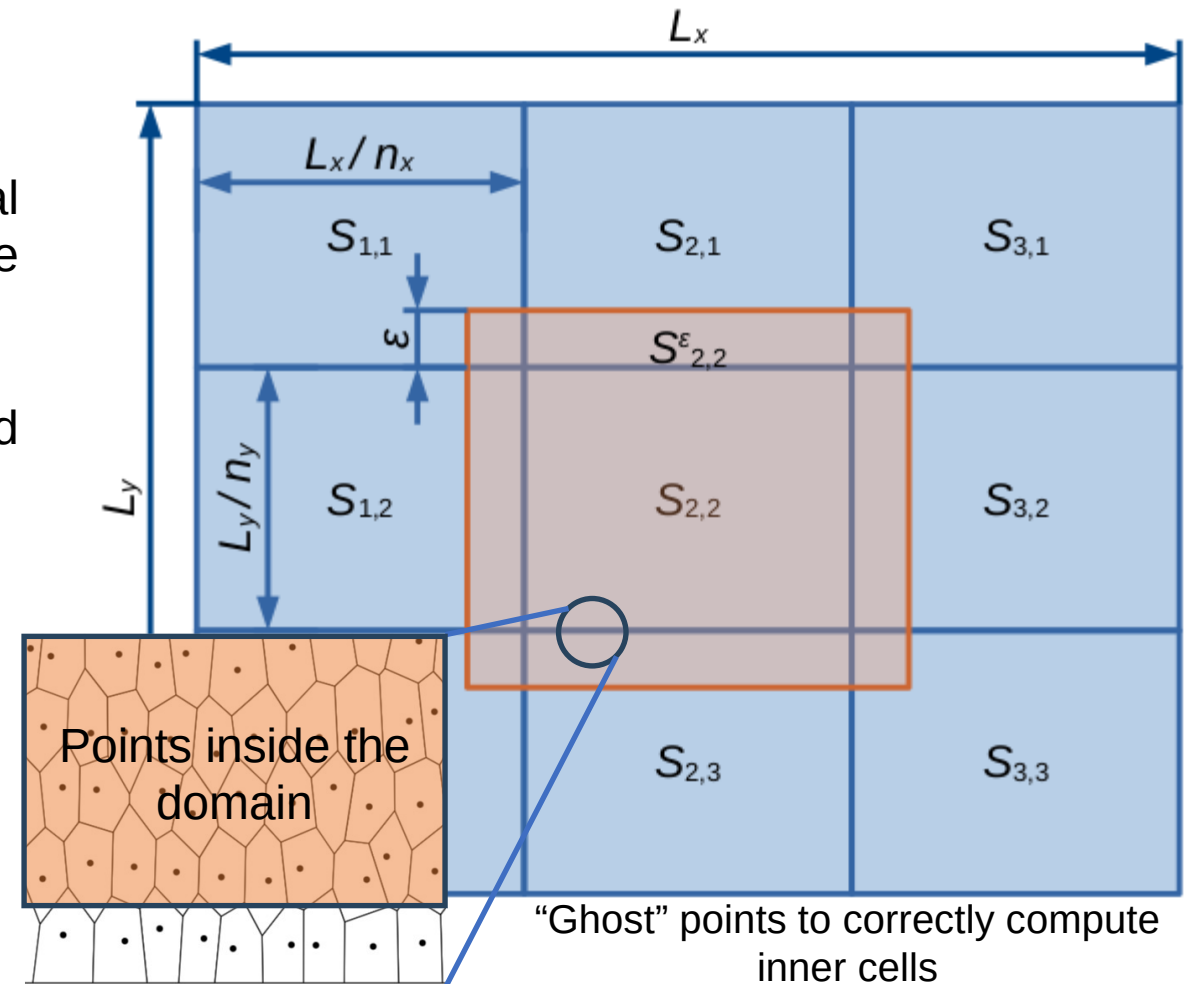
- Particle density is nearly uniformly distributed
- All nearest neighbors lie within distance ε from central particle which is close to the average of interparticle distance d_{ave} : $\varepsilon = \alpha d_{ave}$ where $\alpha \in (1, 5]$

An Orthogonal Box is divided into $n_x \times n_y \times n_z$ equal-sized subdomains which we denote as $\mathbf{S}_{i,j,k}$.

To include particle neighbor's each subdomain is extended a distance ε .

Extended subdomains $\mathbf{S}^{\varepsilon}_{i,j,k}$ are processed in parallel

Load balancing: generate more subdomains than there are hardware threads





Parallel Voronoi tessellation algorithm

Input:

L_x, L_y, L_z — box size in 3 dimensions;

n_x, n_y, n_z — number of blocks to split each dimension into;

ε — skin distance;

$P = \{r_p \in (0, L_x) \times (0, L_y) \times (0, L_z), p = 1, \dots, N_{part}\}$ — array of particle coordinates;

Output:

Voronoi tessellation of the box $(0, L_x) \times (0, L_y) \times (0, L_z)$ by particles $p \in P$



PARALLELIZATION STRATEGY

begin

Generate an $n_x \times n_y \times n_z$ array of subdomains $S = \{S_{i,j,k}^\varepsilon\}$;

Generate an $n_x \times n_y \times n_z$ array of partial tessellations $V = \{V_{i,j,k}\}$ such
that $S_{i,j,k}^\varepsilon$ defines the bounding box for $V_{i,j,k}$;

// Sort particles into subdomains

foreach r_p in P **// Serial or parallel**

forall (i, j, k) such that $r_p \in S_{i,j,k}^\varepsilon$

 add r_p to $V_{i,j,k}$ with a numerical ID p

end

end

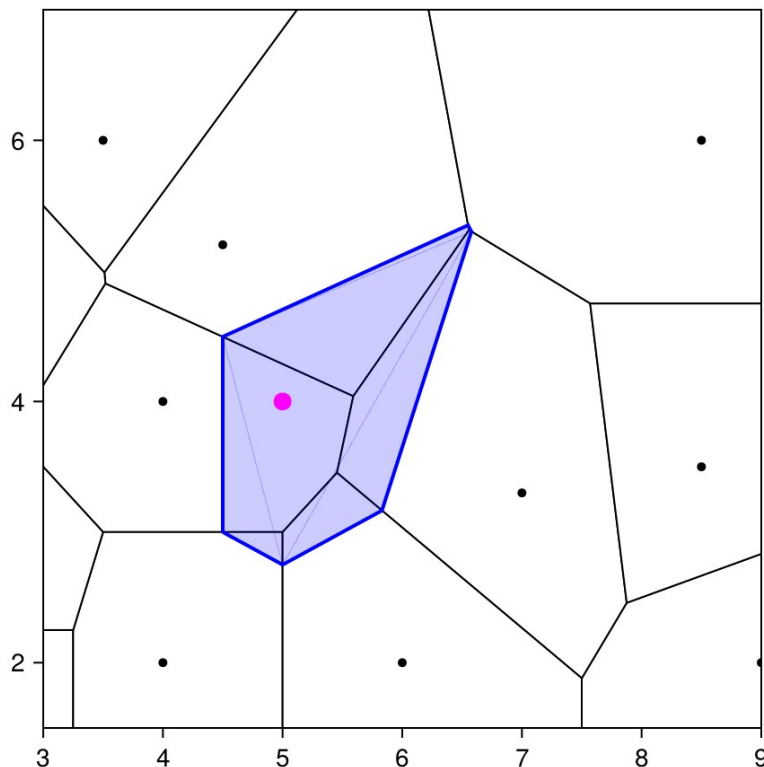
foreach $V_{i,j,k}$ in V **// Parallel**

 Compute the Voronoi tessellation using all particles added to $V_{i,j,k}$;

 Output or analyze the cells for particle IDs $p : r_p \in S_{i,j,k}$

end

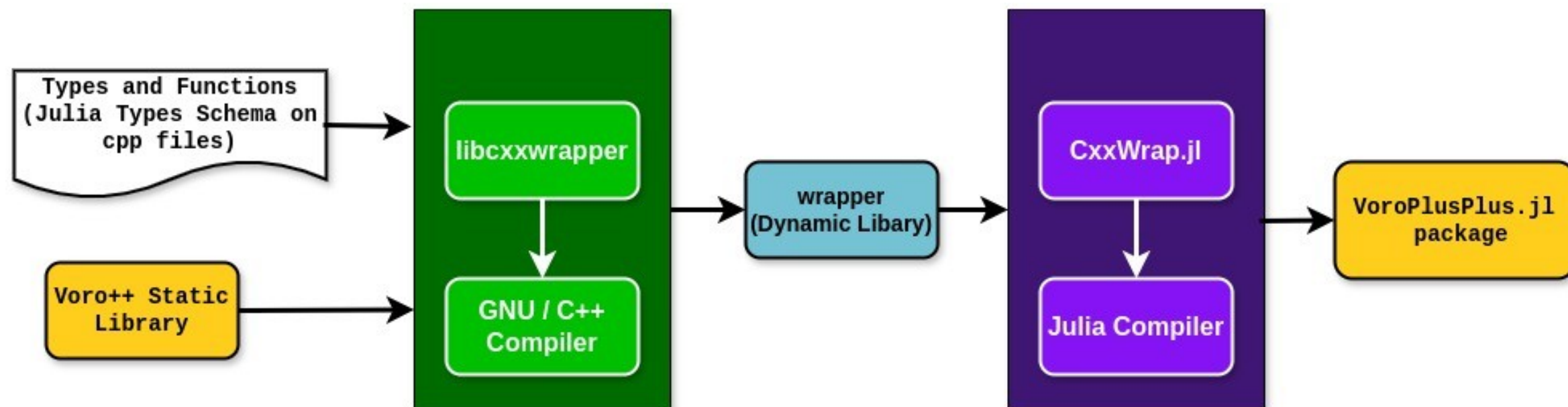
end



1. Introduce a testing point x_0 and calculate the Voronoi tessellation
2. Find the Voronoi cell neighbors for x_0 .
3. Calculate weights (coefficients) for the neighbors
4. Calculate interpolating value f_0 as weighted sum over neighbors

$$f_0 = \sum_{m=1}^M \alpha_m f_m$$

$$\alpha_m = \frac{s_m}{h_m} / \sum_{j=1}^M \frac{s_j}{h_j}$$





Cell Computation – Calculation of tessellation predominates in time of entire process.

As part of this calculation there are tasks that can be made totally in parallel because of their non-deterministic nature.

Serial interface:

All public methods for Container class and VoronoiCell class in Voro++ are exported via wrapper library.

In addition, iteration over the parallel container is possible serially

iterate(vt::VoronoiTessellation) – needed for loops

```
for cell in vt
  # do something – serially
end
```

Parallel interface:

foreach(func, vt::VoronoiTessellation) applies function **func** to each of the cells in tessellation **vt**;

```
foreach(vt) do cell
  # do something – in parallel
end
```

Special case:

mapreduce(func, op, vt::VoronoiTessellation; init) applies function **func** to each of the cells in tessellation **vt** and reduces the results using operator **op**.

```
mapreduce(+, vt; init=0.0) do cell
  volume(cell)
end
```



Implementation of a multi-threaded shared-memory

Within the Julia program calling an external library, such parallelism of computations may be achieved in two ways:

1. First, the external library can be parallelized using the language it is written in ("foreign" multi-threading) .
2. Second, the Julia code may be parallelized using Julia multi-threading library ("native" multi-threading) .

* **In theory**, both parallelization strategies may coexist in one program.

* **In practice**, to avoid CPU oversubscription, all parallel tasks must be managed by the same scheduler.

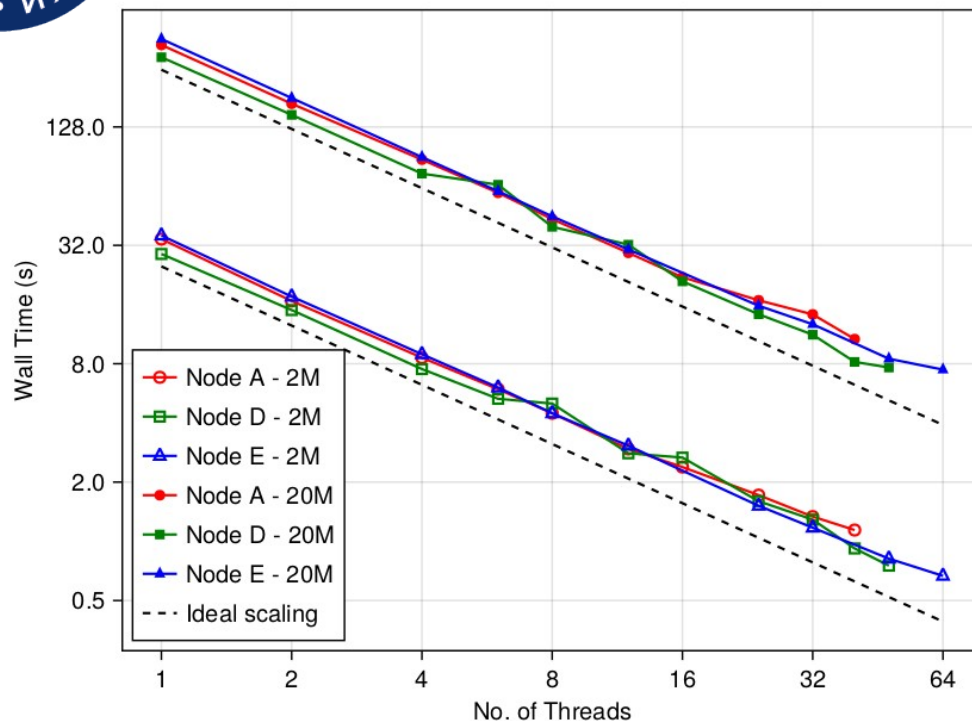


SYSTEM CHARACTERISTICS FROM “cHARISMa” HPC

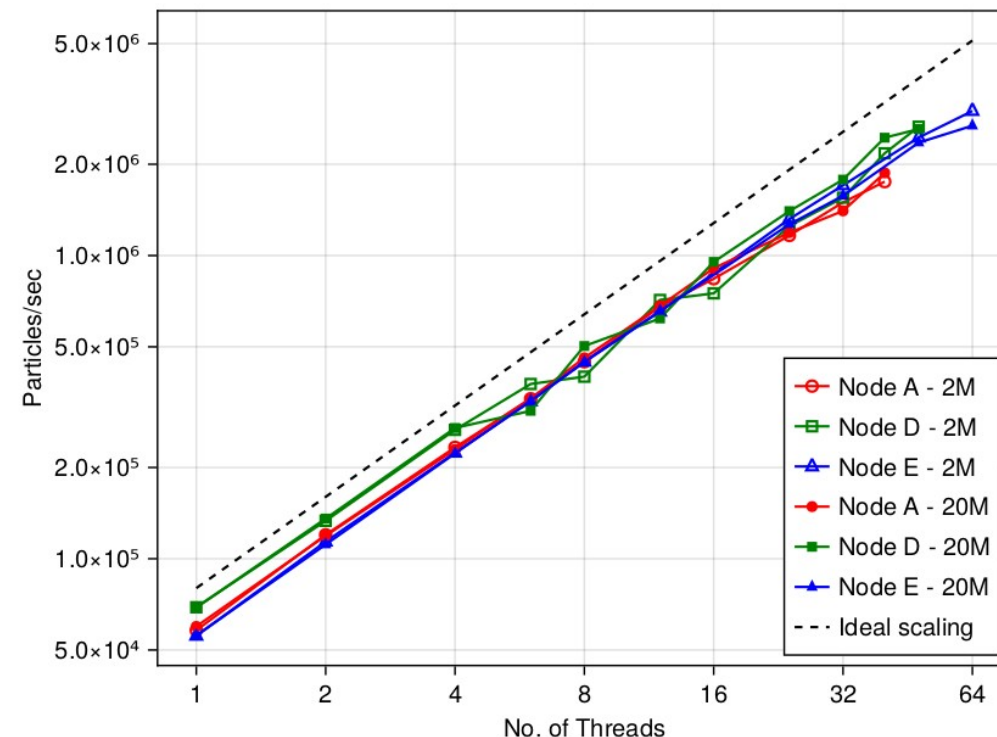
HPC System	
Operating System	GNU / Linux CentOS
C++ Compiler	GNU Compiler Collection 12.1
Voro++ Version	0.4.6
Julia Version	1.10.8

HPC Nodes			
Node	Type A	Type D	Type E
CPU Type	2× Intel Xeon Gold 6152 2.1-3.7 GHz	2× Intel Xeon Gold 6248R 3.0-4.0 GHz	2× AMD EPYC 7702 2.0-3.35 GHz
Max. CPU Cores	44	48	128
RAM	768 GB DDR4	768 GB DDR4	1 TB DDR4

BENCHMARKING: UNIFORM DISTRIBUTION

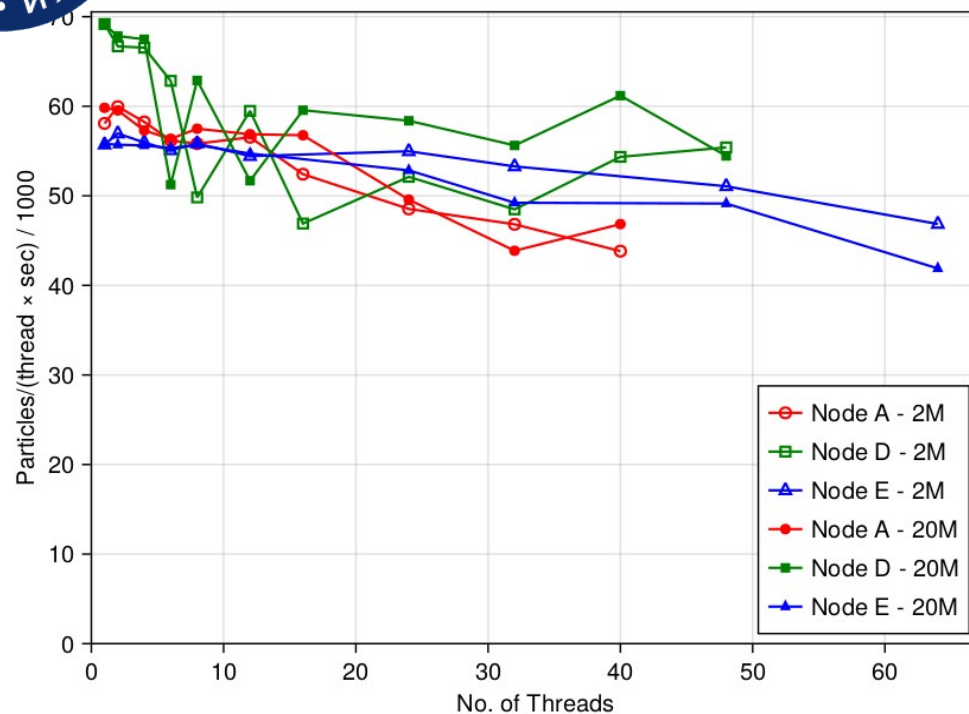


Computation time against number of threads on different computational nodes, with 2 million and 20 million particles problem. The dashed lines corresponding to $1/n_{\text{threads}}$ ideal case.

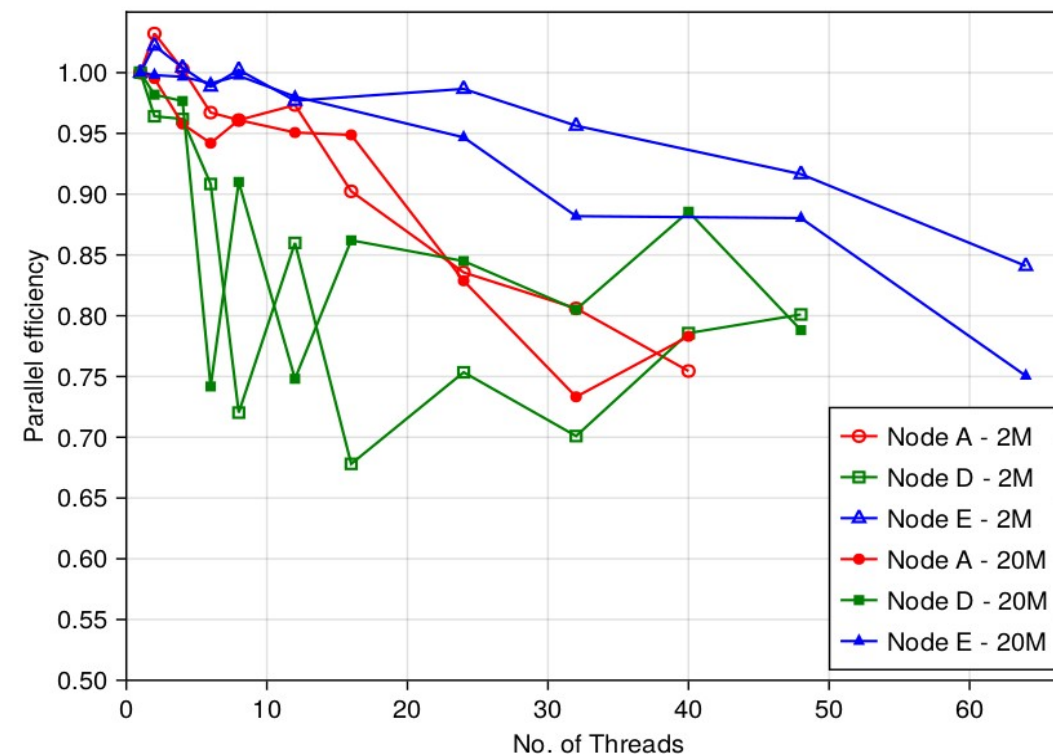


Number of processed particles per second against number of threads on different computational nodes, with 2 million and 20 million particles problem.

BENCHMARKING: UNIFORM DISTRIBUTION

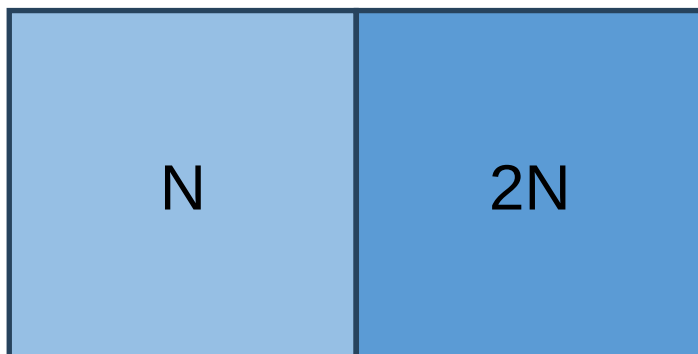


Number of processed particles per second per thread against number of threads on different computational nodes, with 2 million and 20 million particles problem.



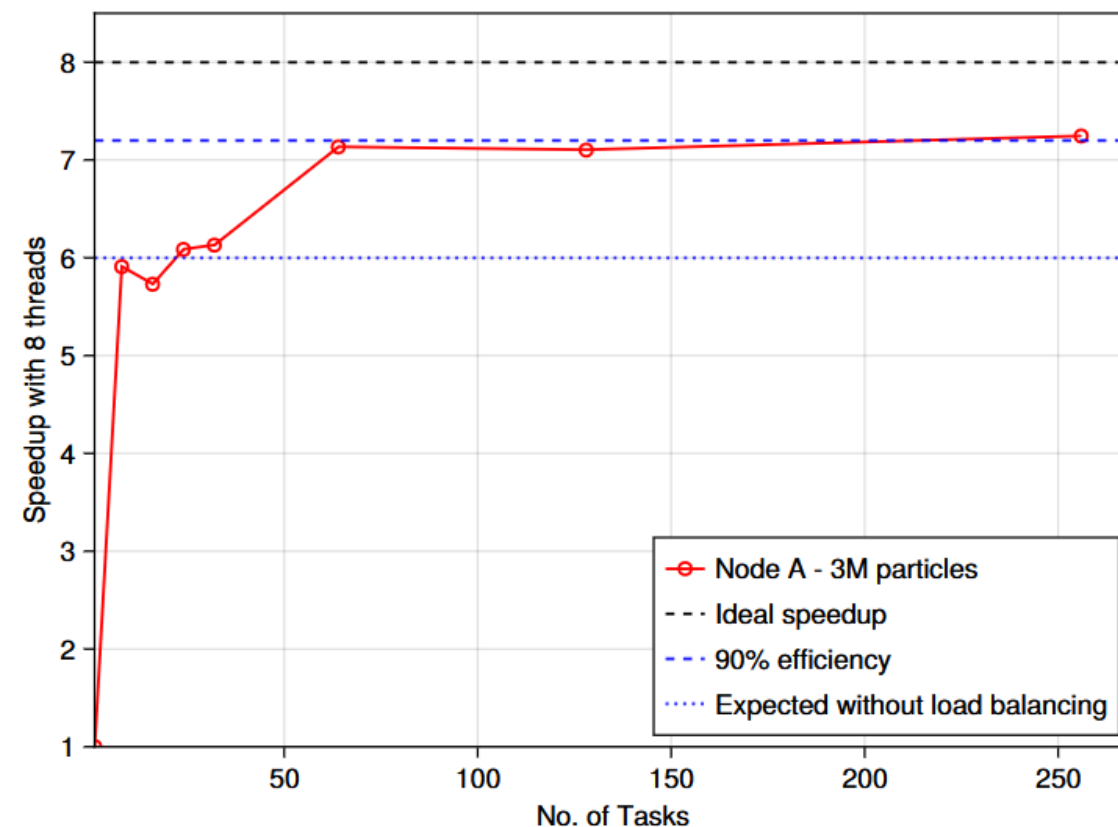
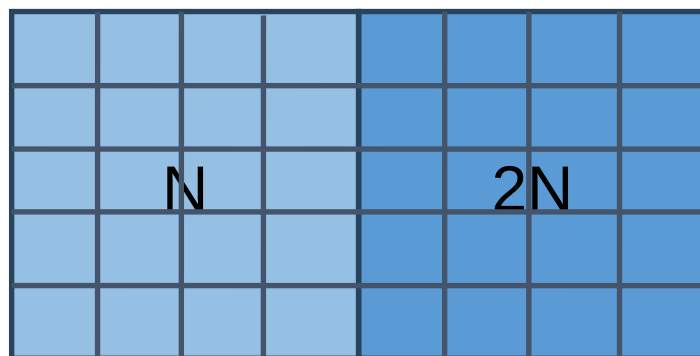
Parallel efficiency against number of threads on different computational nodes, with 2 million and 20 million particles problem.

BENCHMARKING: NON-UNIFORM DISTRIBUTION



Half of the domain is filled with double particle density

Workload balancing: Create more subdomains than hardware threads



Parallel speedup on 8 threads with different number of tasks (subdomains) for a 3 million-particle problem.



Future goals:

1. Parallelization for non-rectangular container shapes.
2. Support for walls.
3. Explore load balancing strategies



CONCLUSIONS

- A parallel implementation of Voronoi tessellation in Julia Language is used for a domain decomposition strategy, where the initial problem is divided into overlapping subdomains, and a serial algorithm is used for the solution in each of the subdomains.
- The implementation uses Voro++ library as a serial engine and Julia threading system for parallel task scheduling. The library interface is exported in Julia using the CxxWrap.jl library.
- The applicability of Voronoi tessellations aimed to Natural Neighbor Interpolation scheme reflects the possibility to develop an algorithm that generates density maps in molecular dynamics for the first time using this method.



Your attention is appreciated!