Karlsruhe Institute of Technology

SCC

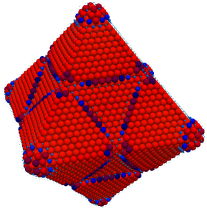
The NAStJA Framework and the Design for High-performance Computing Applications

GridKa School 2019

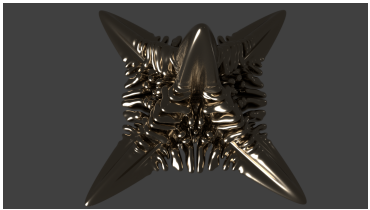
Marco Berghoff, marco.berghoff@kit.edu

STEINBUCH CENTRE FOR COMPUTING

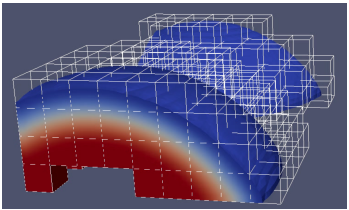
The NAStJA Framework – Applications from nm to cm



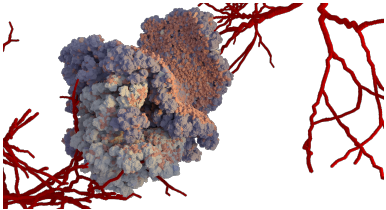
Atomistic phase-field crystal



Phase-field method: dendritic solidification [1]



Phase-field method: wetting phenomena

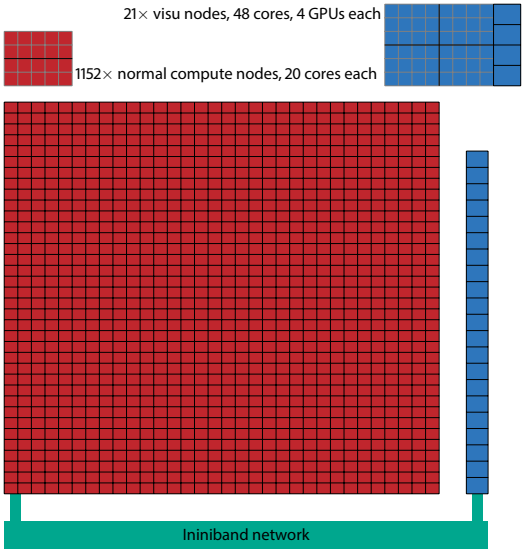


Cellular Potts-model: cancer evolution [2]

High-performance Computing

- Example ForHLR II
- Many compute nodes
- Nodes connected by a high throughput (2.5 Gbit/s) and low latency (0.5 µs) InfiniBand network

Typically, one large problem is distributed to multiple nodes.



The Goal

- Best possible performance
- Olympic motto: faster, higher, stronger

The Goal

- Best possible performance
- Olympic motto: faster, higher, stronger

Enable better science

- More details
- Higher resolution
- More complex models
- More simulations (parameter scans)

7 Point to Increase Performance

- Reality/Experiments
- Physical Parameter
- Mathematical Model
- Numerical Scheme
- Application Program
- Parallel Computing (MPI, …)
- Hardware Architecture

[3]

7 Point to Increase Performance

Reality/Experiments

Physical Parameter

Mathematical Model

Numerical Scheme

Application Program

Parallel Computing (MPI, …)

Hardware Architecture

- What do we want to explore?
- The classical modeling; what can be simplified?

7 Point to Increase Performance

- Reality/Experiments
- **Physical Parameter**
- Mathematical Model
- Numerical Scheme
- Application Program
- Parallel Computing (MPI, …)
- Hardware Architecture

- Make it linear, constant, or reduce them.

7 Point to Increase Performance

- Reality/Experiments
- Physical Parameter
- **Mathematical Model**
- Numerical Scheme
- Application Program
- Parallel Computing (MPI, …)
- Hardware Architecture

- Use a feasible model.
- Atomistic (MD) vs. continuum model

7 Point to Increase Performance

- Reality/Experiments
- Physical Parameter
- Mathematical Model
- **Numerical Scheme**
- Application Program
- Parallel Computing (MPI, …)
- Hardware Architecture

- Forward Euler is simple but has a limited time-step width.
- Higher-order methods need more computational power, needs more data fields

7 Point to Increase Performance

- Reality/Experiments
- Physical Parameter
- Mathematical Model
- Numerical Scheme
- **Application Program**
- Parallel Computing (MPI, …)
- Hardware Architecture

- Often a constrained.
- Not everything is rewritten, old general-purpose code is used.

7 Point to Increase Performance

Reality/Experiments

Physical Parameter

Mathematical Model

Numerical Scheme

Application Program

Parallel Computing (MPI, …)

Hardware Architecture

- MPI is “the standard” for distributed memory.
- It worth to have a hybrid MPI+OpenMP or using GPU to parallelize?

7 Point to Increase Performance

Reality/Experiments

Physical Parameter

Mathematical Model

Numerical Scheme

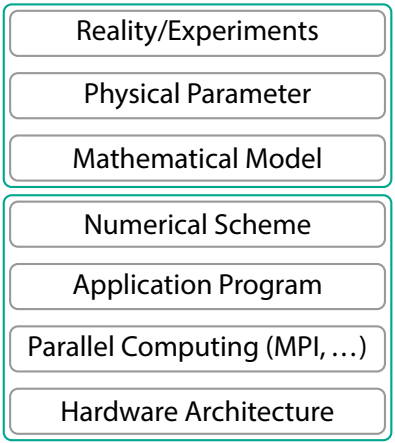
Application Program

Parallel Computing (MPI, …)

Hardware Architecture

- Adapt to processor, instruction set extensions SSE, AVX, AVX2, AVX256. Adapt to cache.

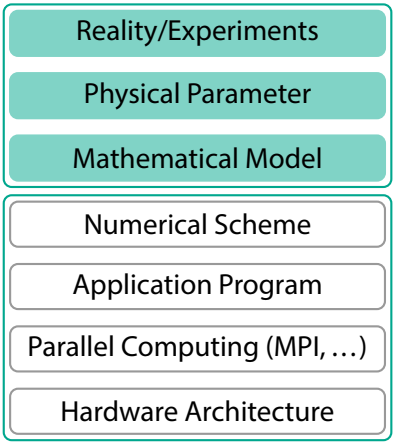
7 Point to Increase Performance



- First group is hardly application-driven.
- Second group is computational performance-driven.

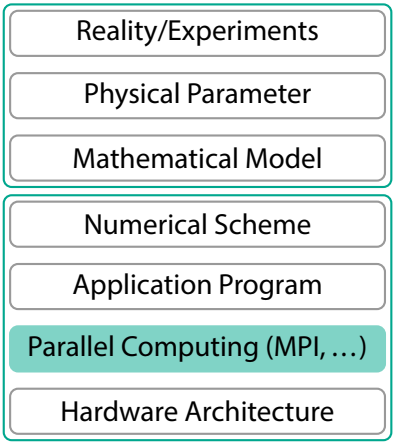
The best performance can only be reached when each point is optimized.

Meaning for the Design / Outline



Omitting Unnecessary Calculations

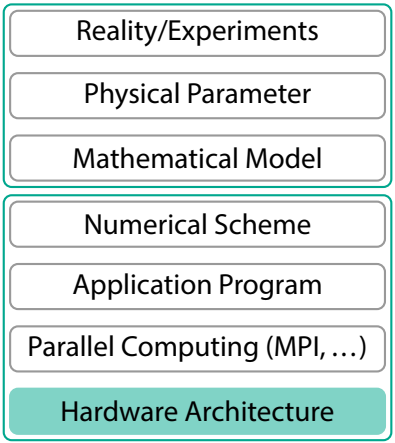
Meaning for the Design / Outline



Omitting Unnecessary Calculations

Scalable Parallel Communication

Meaning for the Design / Outline



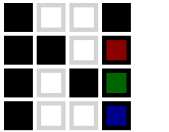
Omitting Unnecessary Calculations

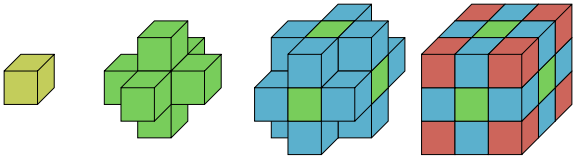
Scalable Parallel Communication

High Node-level Performance

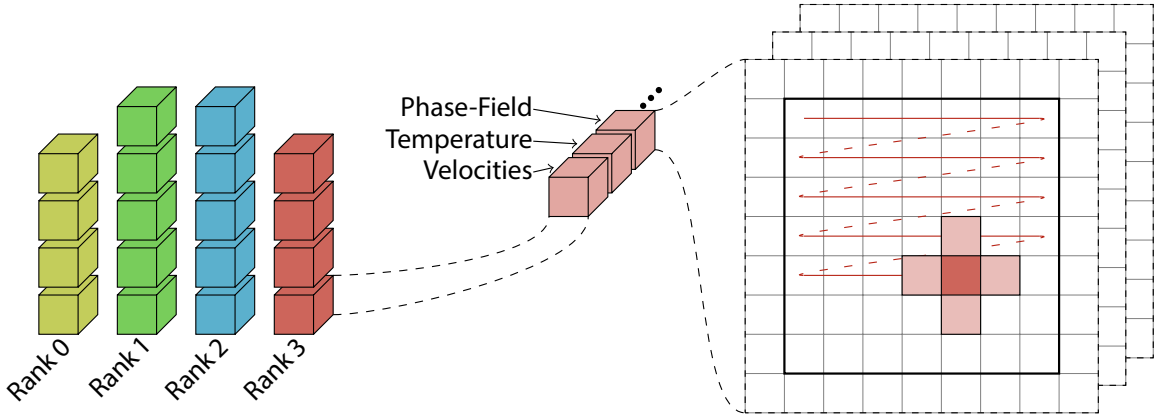
The NAStJA-Framework

- NAStJA: Neoteric Autonomous Stencil code for Jolly Algorithms
- Data: regular grid
- Stencil calculations: read neighbors and write to the center for each grid point
- Block Structured Grid: basis for decomposition

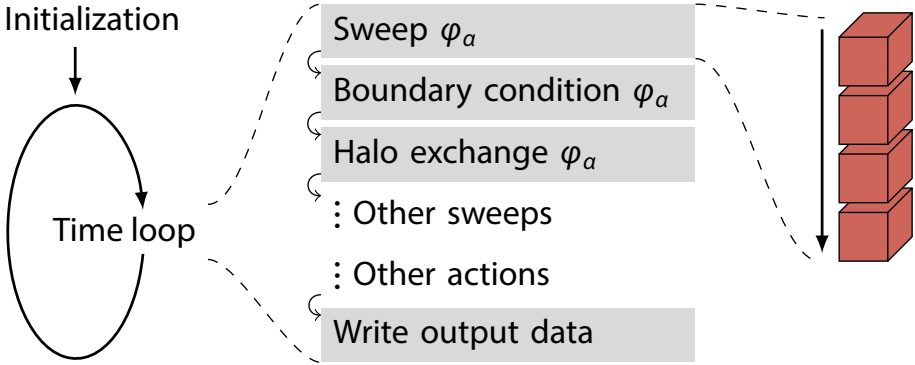




Block Structured Grid – Distribution of Blocks



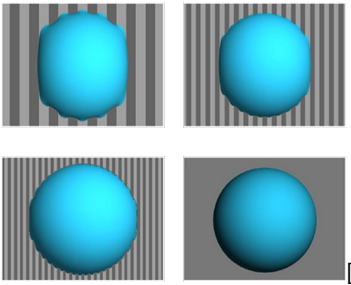
Block Structured Grid – Calculation



Omitting Unnecessary Calculations

Motivation: Droplets on Structured Surface

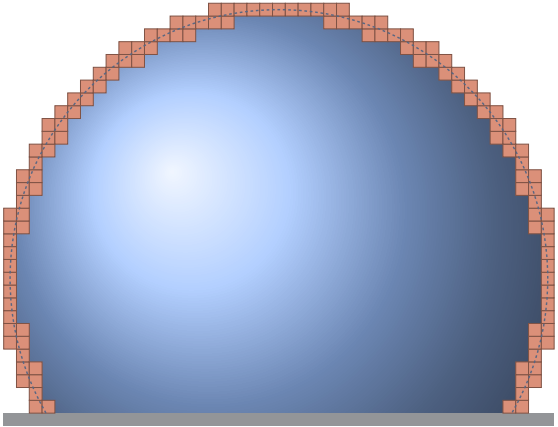
- Rain droplet $\sim$ **3 mm**
- Chemically structured surfaces with **700 lamellar** hydrophobic and hydrophilic stripes
- Phase-field method: **10 grid points** for an interface liquid–air
- Bulk on lamella $\longrightarrow$ resolution of lamella 15 grid points
- $\longrightarrow$ lamella 5 µm $\longrightarrow$ **10^{12} grid points**
- Droplets on spider's web threads 0.5 – 5 µm



[4]

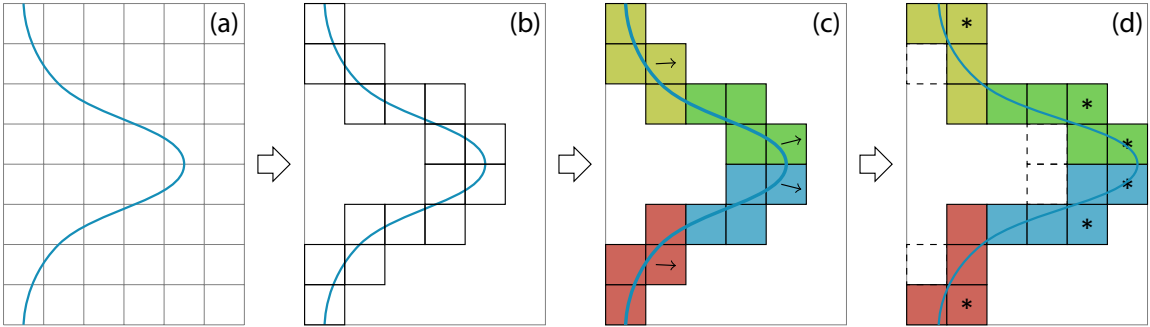
[4]

Only the Interface is Needed

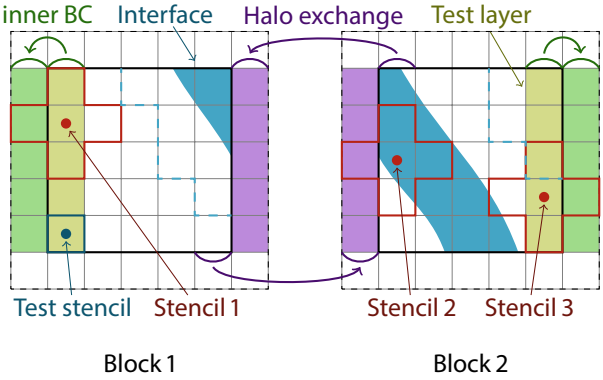


- 2D cut: 177 of 1462 blocks
- only 3.5 % of blocks are needed (edge length 25)

Block Structure Grid – Distribution



Dynamic Block Adaption – Blocks



Moving: Maximal 1 grid point per time-step
Creating new blocks ⟶ Activate halo exchange
How do we know where the neighboring blocks are?

Scalable Parallel Communication

All-to-All

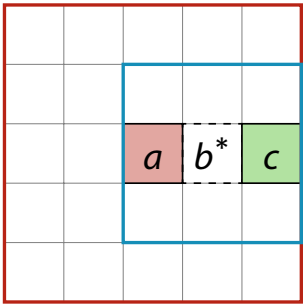
One-to-All, All-to-One

Point-to-Point

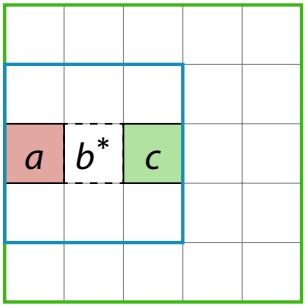
no communication

$\mathcal{O}(N^2)$	matrix transpose, fast Fourier-transformation, residuum calculations
$\mathcal{O}(N)$	master–worker, broadcasting
$\mathcal{O}(1)$	halo exchange
0	best communication (not HPC anymore)

Dynamic Block Adaption – Neighborhood: 125 Blocks

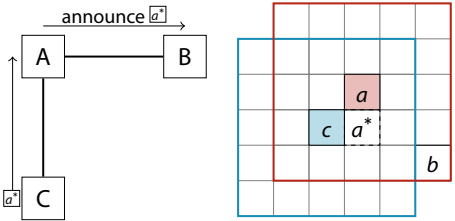


local neighborhood of block *a*



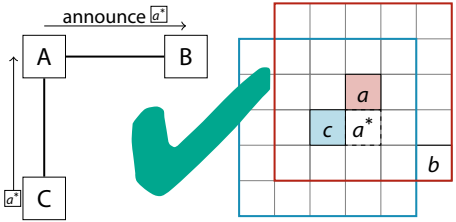
local neighborhood of block *c*

Dynamic Block Adaption – Cases

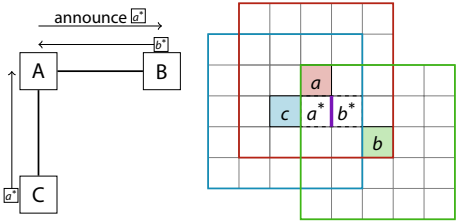


Neighbor communication

Dynamic Block Adaption – Cases

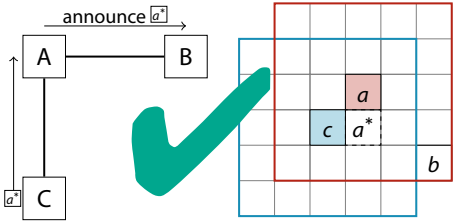


Neighbor communication

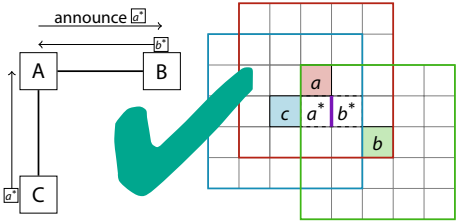


Forward communication

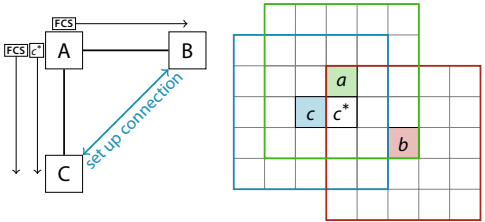
Dynamic Block Adaption – Cases



Neighbor communication

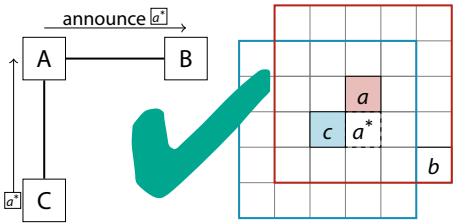


Forward communication

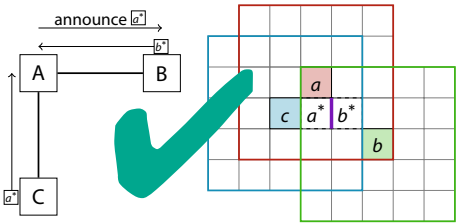


New neighbor

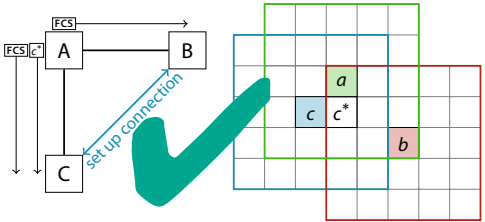
Dynamic Block Adaption – Cases



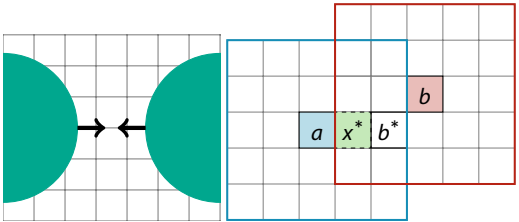
Neighbor communication



Forward communication



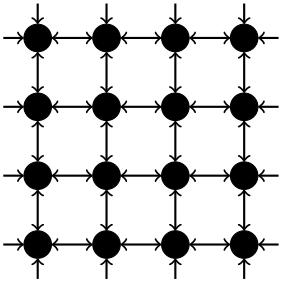
New neighbor



Disjoint groups

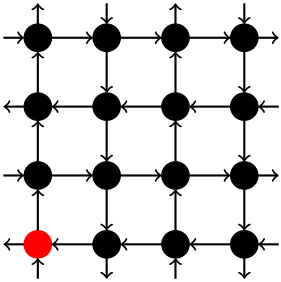
Global Information Exchange

- No collective communication
- Only Point-to-Point communications
- Multiple hops to spread the information
- Simple Network: torus, higher dimension to decrease the diameter

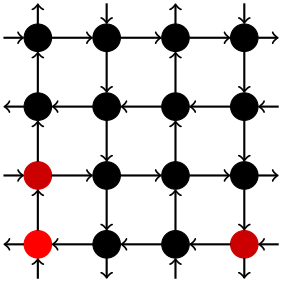


Diameter = the number of hops needed to spread the information globally

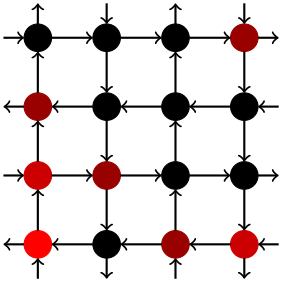
Global Information Exchange – Manhattan Street Network



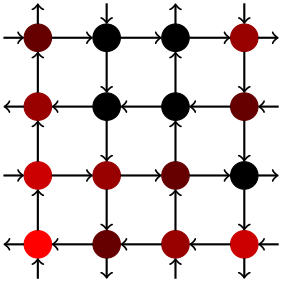
Global Information Exchange – Manhattan Street Network



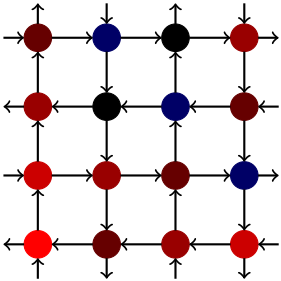
Global Information Exchange – Manhattan Street Network



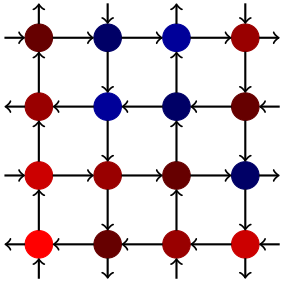
Global Information Exchange – Manhattan Street Network



Global Information Exchange – Manhattan Street Network

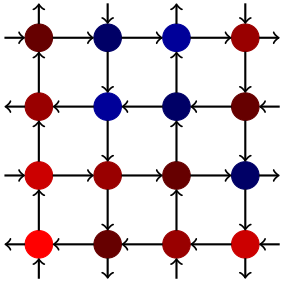


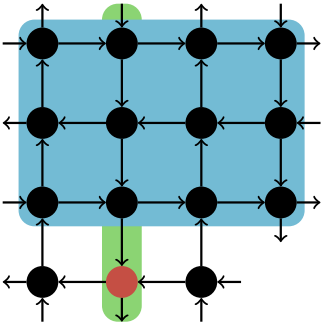
Global Information Exchange – Manhattan Street Network



$$\text{Diameter} = \sum_{i=0}^{\text{Dim}-1} \frac{\text{Nodes in } i\text{-direction}}{2} + \{0,1\}$$

Global Information Exchange – Manhattan Street Network

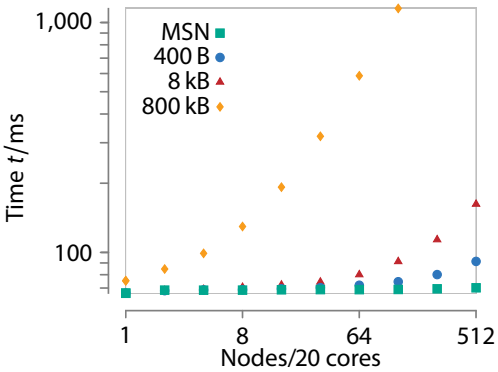


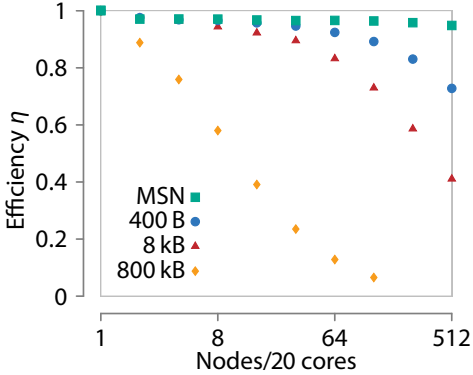


$$\text{Diameter} = \sum_{i=0}^{\text{Dim}-1} \frac{\text{Nodes in } i\text{-direction}}{2} + \{0,1,2\} \qquad \text{Diameter} \approx \frac{\text{Dim}}{2} \sqrt[\text{Dim}]{\text{Nodes}}$$

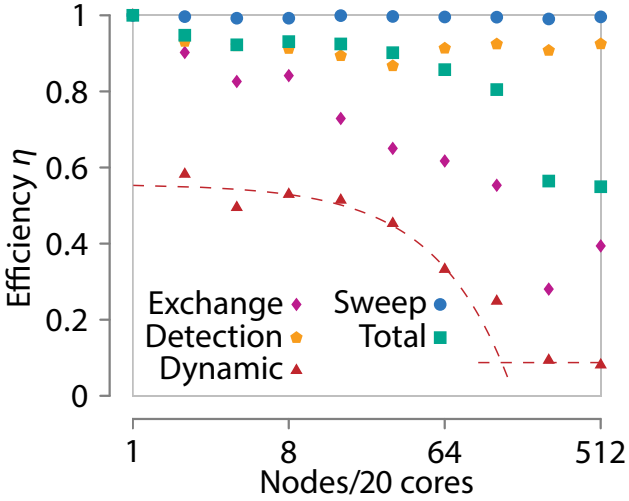
Use higher dimension/degree for a smaler diameter.

Global Information Exchange – Efficiency





Overall Efficiency for Dynamic Blocks

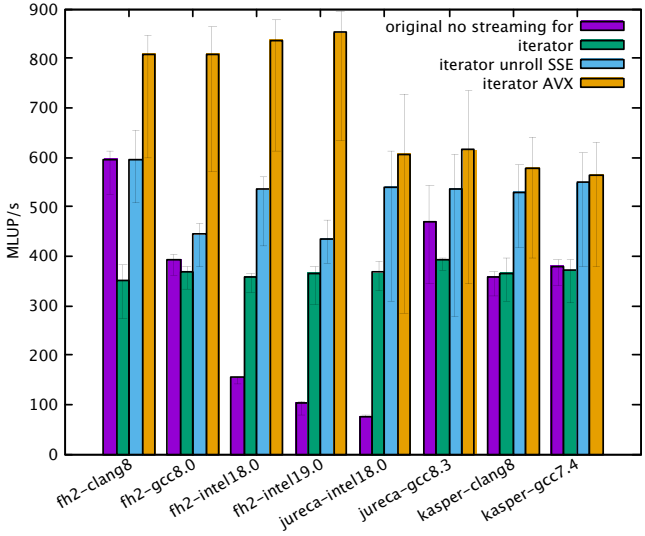


Node-level Performance

Goal: reache nearly peak-performance on core or node

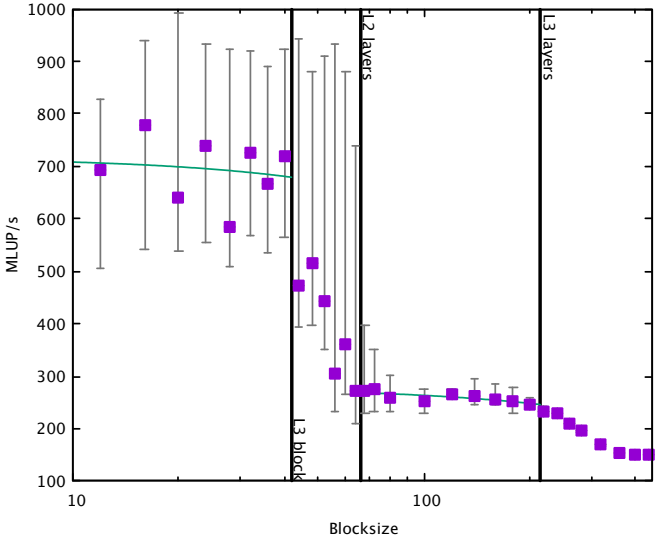
- Only $5 - 10\%$ peak-performance:
  without 2 FMAs, 2 instructions each, 4 doubles (AVX)
- Adaptive mesh refinement (AMR), etc. on block level
- No index calculation – just stencil streaming
- The calculation is done in a sweep `for(voxel:field) { //do ... }`
- Easy replacement of sweeps with optimized versions

Streaming



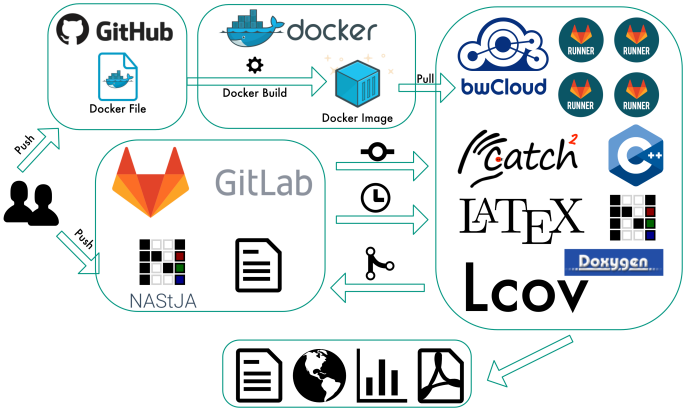
- MLUP/s: million lattice updates per seconds (more is better)
- `for z, y, x` can be slow
- Iterator over field
- Easy to enable SSE or AVX optimization

Cache Usage



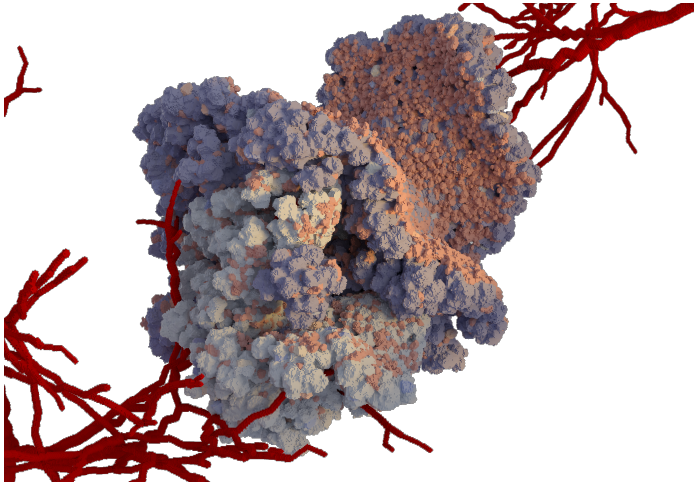
- L3 cache
  - Block $\sim$ 700 MLUP/s
  - 3 read + 1 write layers
    $\sim$ 260 MLUP/s
- L2 cache
  - 3 + 1 layers

Testing Environment



- Unit tests
- Simulation test – compared to md5sums
- Optimized sweeps can be compared to unoptimized

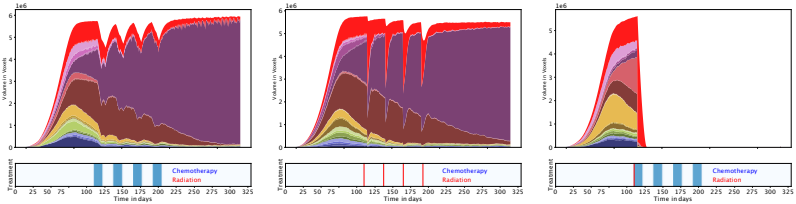
Application: Cancer



[2]

- Bloodvessel, Tumor cell types, apoptotic cells
- Surrounding tissue is suppressed
- $1000 \times 1000 \times 1000$ voxel $\sim 1\,\mathrm{mm}^3$
- 1 million biological cells with 1000 voxel each
- 250 000 Monte Carlo sweeps on 1000 ranks with 24 h runtime
- $\sim$ 6 month simulated time

Application: Cancer Treatments



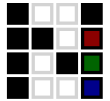
- Chemotherapy
  - Cytostatic agent is deployed by blood vessels
  - Local alteration of division rates
- Radiation therapy
  - Global lowering of division rates
  - Immediate apoptosis

[2]

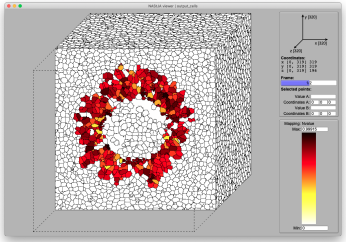
Conclude

- Omitting Unnecessary Calculations
- Scalable Parallel Communication
- High Node-level Performance

- NAStJA open source soon™
- NAStJAviewer



NAStJA



https://gitlab.com/nastja/viewer

Credits: [1] N. Pfisterer, [2] J. Rosenbauer, [3] J. Hötzer, [4] M. Ben Said