University of Stuttgart
Institute of Aerospace Thermodynamics

HLRS
High-Performance Computing Center | Stuttgart

A method to reduce load imbalances in simulations of phase change processes with FS3D

Johannes Müller
Philipp Offenhäuser
Martin Reitzle

Workshop on Sustained Simulation Performance
HLRS, Stuttgart, Germany
October 9th and 10th, 2018

Introduction

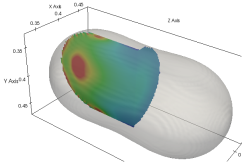
Institute of Aerospace Thermodynamics

- Droplet Dynamics Group
- Simulations of Multiphase flows with Free Surface 3D (FS3D)

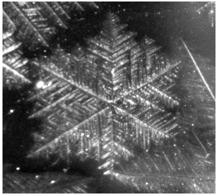
Dynamic of Free Surface

Phase changes

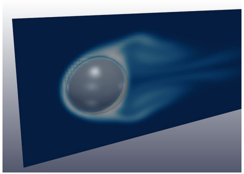
Droplet Splashing



Droplet Dynamics



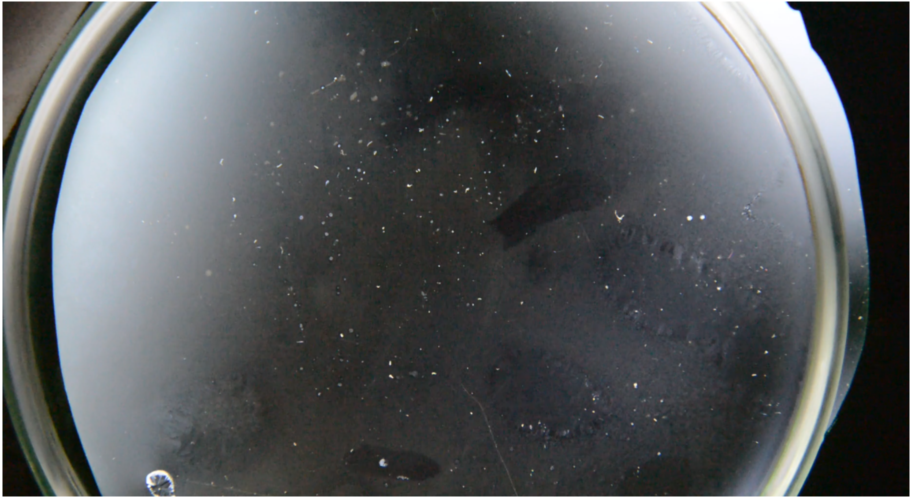
Freezing



Evaporation

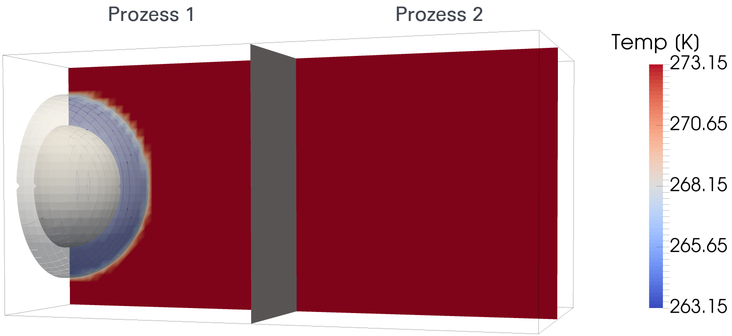
Motivation

Phase change process: supercooled water to ice



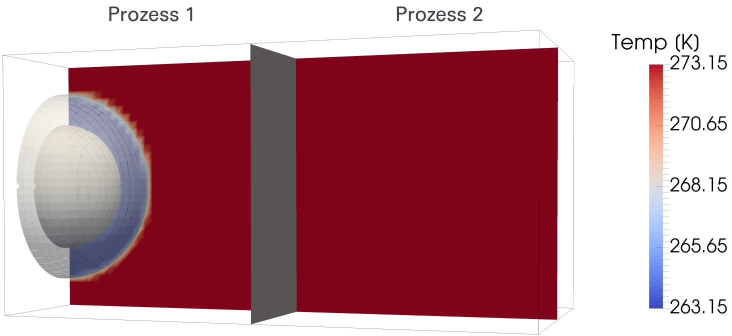
Motivation

Numerical simulation of supercooled droplet

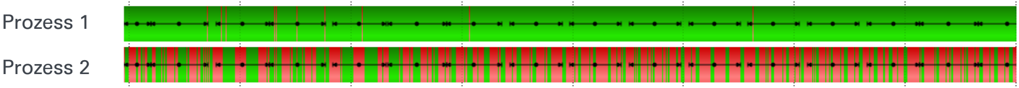


Motivation

Numerical simulation of supercooled droplet



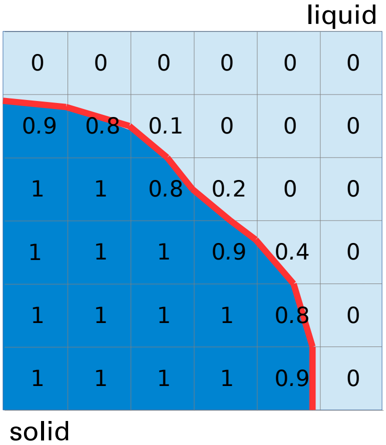
Load Imbalance:



Outline

Solidification simulation

Volume of Fluid method (VOF)

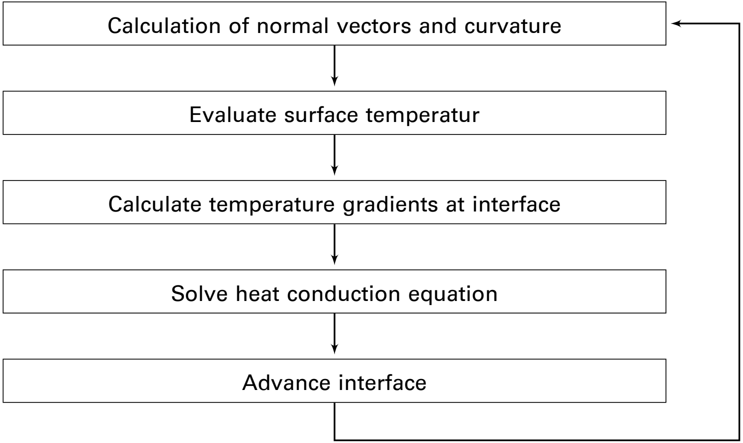


Fractional volume:

$$f(\vec{x},t) = \frac{V_{solid}}{V_{cell}} = \begin{cases} 0 & \text{in the liquid phase} \\ 0 < f < 1 & \text{in the boundary cells} \\ 1 & \text{in the solid phase} \end{cases}$$

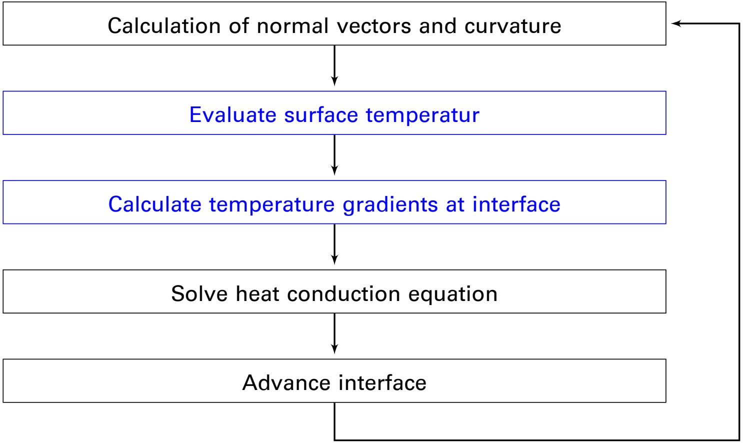
Solidification simulation

Solidification model



Solidification simulation

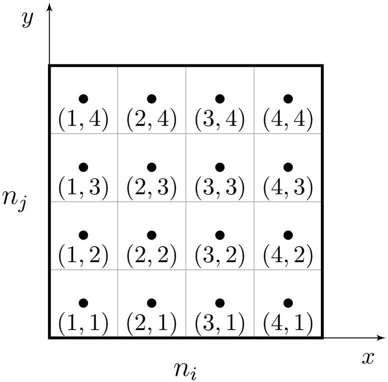
Solidification model



Solidification simulation

Parallelization

Discretized computational domain: Indexing of control volumina

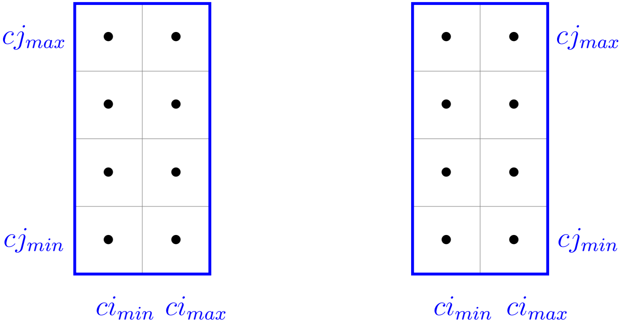


- Structured, equidistant grid
- Data structure is based on 3-dimensional arrays
- Decomposition in contiguous subarrays

Solidification simulation

Parallelization

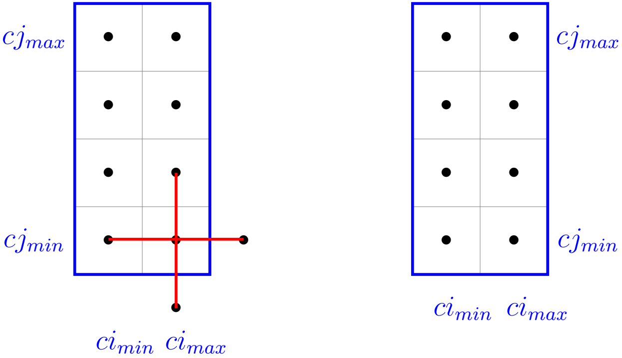
Decomposition of computational domain in subdomains



Solidification simulation

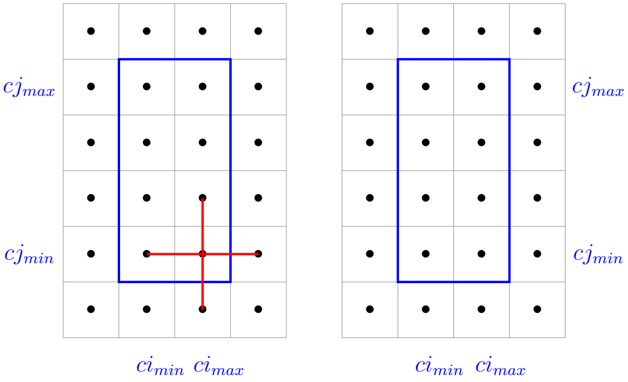
Parallelization

Evaluation of a stencil



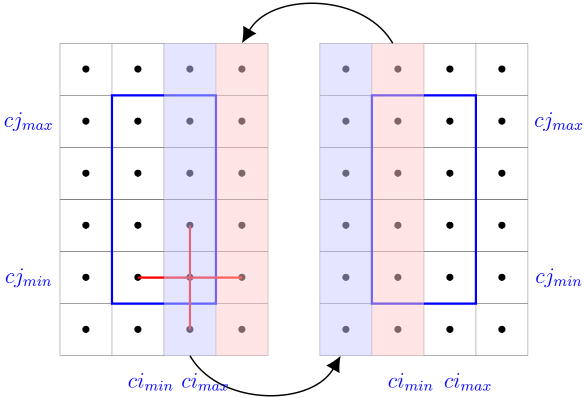
Solidification simulation
Parallelization

Introduction of ghost cells



Solidification simulation
Parallelization

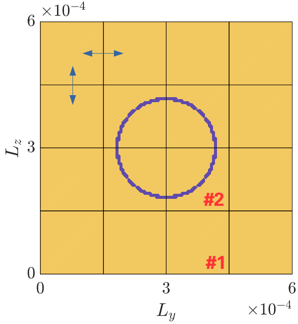
Data exchange between domains



Solidification simulation

Load Imbalance

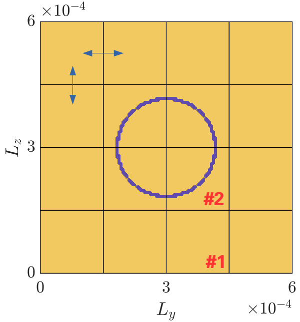
Symmetrical domain decomposition



Solidification simulation

Load Imbalance

Symmetrical domain decomposition



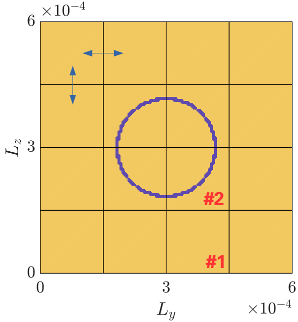
Additional operations in domains with surface cells

```
SUBROUTINE energy jump solid
Communication
t_R_start = MPI_Wtime()
loop k = 1,NK
    loop j = 1,NJ
        loop i = 1,NI
            if 0 < f(i,j,k) < 1 then
                Calculation
            end if
        end loop
    end loop
end loop
t_R_end = MPI_Wtime()
Communication
```

Solidification simulation

Load Imbalance

Symmetrical domain decomposition



Additional operations in domains with surface cells

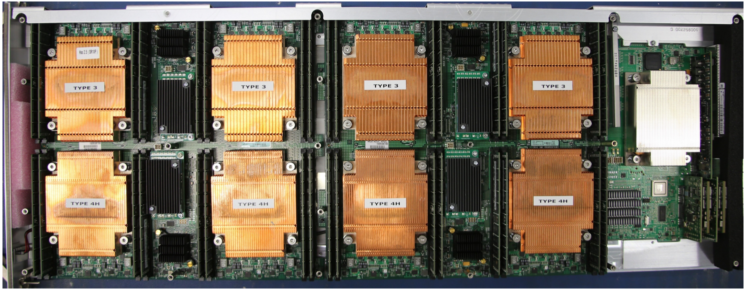
```
SUBROUTINE energy jump solid
Communication
t_R_start = MPI_Wtime()
loop k = 1,NK
    loop j = 1,NJ
        loop i = 1,NI
            if 0 < f(i,j,k) < 1 then
                Calculation
            end if
        end loop
    end loop
end loop
t_R_end = MPI_Wtime()
Communication
```

Work Load Imbalance:

$$\Delta t_2 > \Delta t_1 \quad (1)$$

Load-Balancing method
Optimal Work Load

Compute Blade of Cray XC40: 4 compute nodes with 2 processors each containing 12 cores.



Optimal workload per core

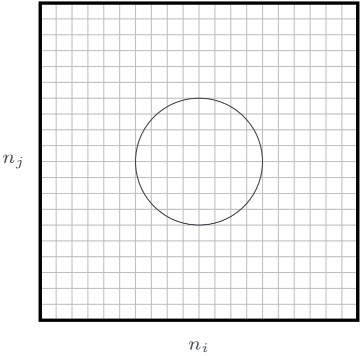
$$L^{opt} = L^{mean} = \frac{\sum_{n=0}^{p-1} L^T}{p_{total}} \quad (2)$$

Workload per Domain

$$L^T = \sum_1^{GN} w \quad (3)$$

Load-Balancing method

Decomposition by Bisection

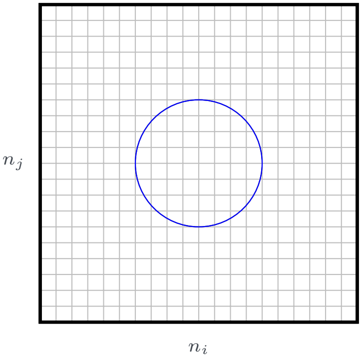



```
assign cells with computational weight
w_e(i, j, k) << w_o(i, j, k)

while D_temp < D_total do
    Find Domain with highest Work Load
    Copy Bisection Coordinates
    Calculate expansion of domain
    Bisection of Domain
        old subdomain c_max halved
        new subdomain c_min corrected
end while
```

Load-Balancing method

Decomposition by Bisection

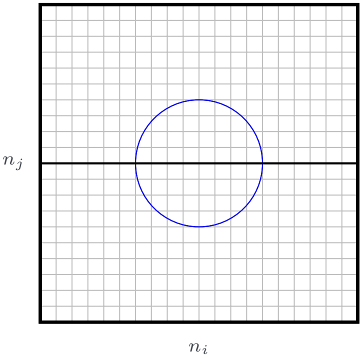


assign cells with computational weight $w_e(i,j,k) << w_o(i,j,k)$

while $D_{temp} < D_{total}$ **do**
  Find Domain with highest Work Load
  Copy Bisection Coordinates
  Calculate expansion of domain
  Bisection of Domain
    old subdomain c_{max} halved
    new subdomain c_{min} corrected
end while

Load-Balancing method

Decomposition by Bisection

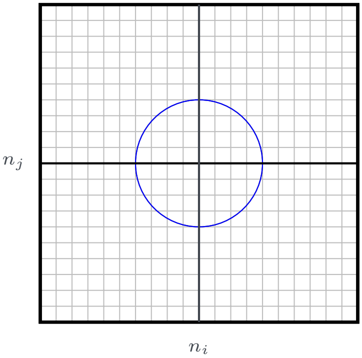


assign cells with computational weight
$w_e(i,j,k) << w_o(i,j,k)$

while $D_{temp} < D_{total}$ **do**
    Find Domain with highest Work Load
    Copy Bisection Coordinates
    Calculate expansion of domain
    Bisection of Domain
        old subdomain c_{max} halved
        new subdomain c_{min} corrected
end while

Load-Balancing method

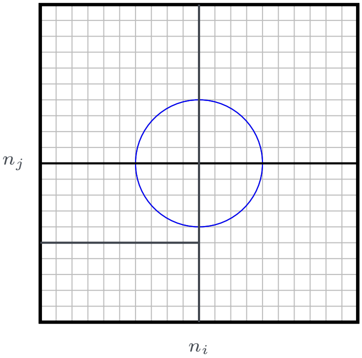
Decomposition by Bisection




```
assign cells with computational weight
    w_e(i, j, k) << w_o(i, j, k)

while D_temp < D_total do
    Find Domain with highest Work Load
    Copy Bisection Coordinates
    Calculate expansion of domain
    Bisection of Domain
        old subdomain c_max halved
        new subdomain c_min corrected
end while
```

Load-Balancing method
Decomposition by Bisection

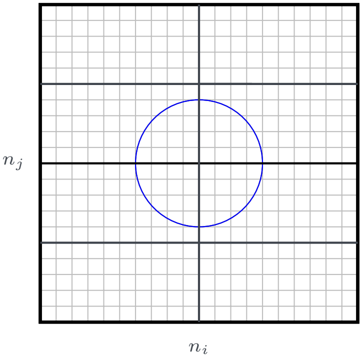


assign cells with computational weight $w_e(i,j,k) << w_o(i,j,k)$

while $D_{temp} < D_{total}$ **do**
    Find Domain with highest Work Load
    Copy Bisection Coordinates
    Calculate expansion of domain
    Bisection of Domain
        old subdomain c_{max} halved
        new subdomain c_{min} corrected
end while

Load-Balancing method

Decomposition by Bisection

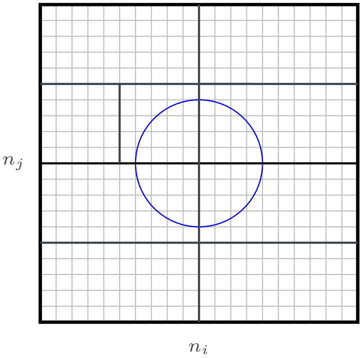


assign cells with computational weight
$w_e(i,j,k) << w_o(i,j,k)$

while $D_{temp} < D_{total}$ **do**
  Find Domain with highest Work Load
  Copy Bisection Coordinates
  Calculate expansion of domain
  Bisection of Domain
    old subdomain c_{max} halved
    new subdomain c_{min} corrected
end while

Load-Balancing method

Decomposition by Bisection

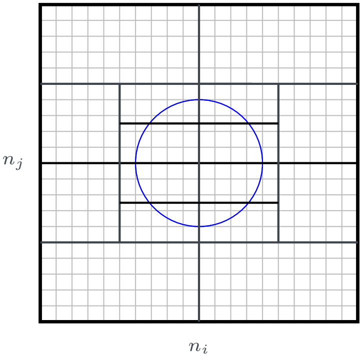


assign cells with computational weight
$w_e(i,j,k) << w_o(i,j,k)$

while $D_{temp} < D_{total}$ **do**
  Find Domain with highest Work Load
  Copy Bisection Coordinates
  Calculate expansion of domain
  Bisection of Domain
    old subdomain c_{max} halved
    new subdomain c_{min} corrected
end while

Load-Balancing method

Decomposition by Bisection

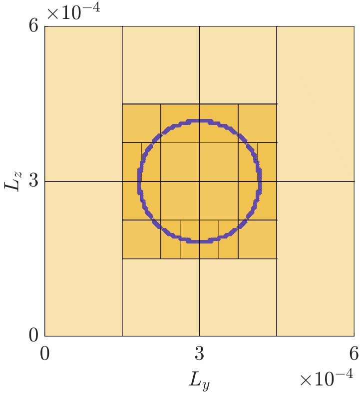


assign cells with computational weight
$w_e(i,j,k) << w_o(i,j,k)$

while $D_{temp} < D_{total}$ **do**
    Find Domain with highest Work Load
    Copy Bisection Coordinates
    Calculate expansion of domain
    Bisection of Domain
        old subdomain c_{max} halved
        new subdomain c_{min} corrected
end while

Load-Balancing method

Load Balanced Domain Decomposition



- Equalized Work Load
- Implementation of process communication:
    - Data exchange to at least 26 neighbors (27-point stencil)
    - Construction of neighborhood (source and destination)
    - Construction of coordinates for send and receive arrays

Results

Testcases

Name	Total number of cells	Initial diameter	% Surface cells
Test Case 1	128^3	$d = d_0$	0.09
Test Case 2	128^3	$d = 2d_0$	0.37

Results

Testcases

Name	Total number of cells	Initial diameter	% Surface cells
Test Case 1	128^3	$d = d_0$	0.09
Test Case 2	128^3	$d = 2d_0$	0.37

Indicator for load imbalance:

$$\text{Imbalance} = \frac{\%\ \text{Surface cells}(P_{max}) - \%\ \text{Surface cells}(P_{mean})}{\%\ \text{Surface cells}(P_{mean})} \tag{4}$$

Results

Domain decompositions for Test Case 1

Load Balancing	LB 1	LB 2	LB 3	LB 4
Weight per cell	$w_e = w_o$	$w_e < w_o$	$w_e << w_o$	$w_e << w_o$
Number of processes with surface cells	2	4	8	
Imbalance	7	3	1	

Results

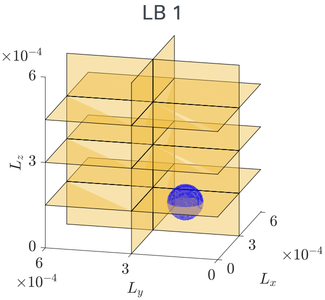
Domain decompositions for Test Case 1

Load Balancing	LB 1	LB 2	LB 3	LB 4
Weight per cell	$w_e = w_o$	$w_e < w_o$	$w_e << w_o$	$w_e << w_o$
Number of processes with surface cells	2	4	8	
Imbalance	7	3	1	

Results

Domain decompositions for Test Case 1

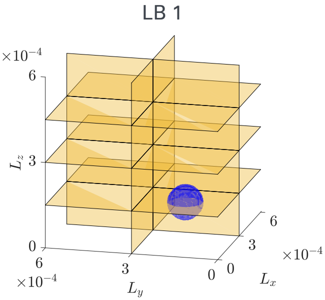
Load Balancing	LB 1	LB 2	LB 3	LB 4
Weight per cell	$w_e = w_o$	$w_e < w_o$	$w_e << w_o$	$w_e << w_o$
Number of processes with surface cells	2	4	8	
Imbalance	7	3	1	

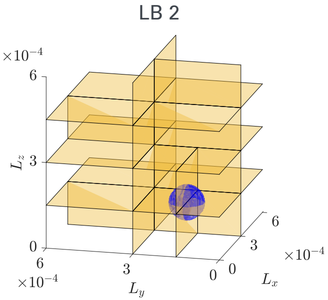


Results

Domain decompositions for Test Case 1

Load Balancing	LB 1	LB 2	LB 3	LB 4
Weight per cell	$w_e = w_o$	$w_e < w_o$	$w_e << w_o$	$w_e << w_o$
Number of processes with surface cells	2	4	8	
Imbalance	7	3	1	



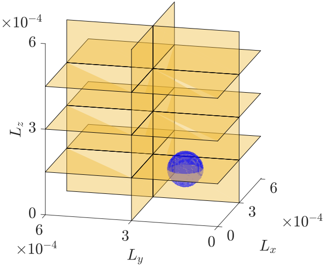


Results

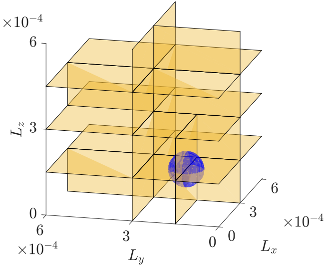
Domain decompositions for Test Case 1

Load Balancing	LB 1	LB 2	LB 3	LB 4
Weight per cell	$w_e = w_o$	$w_e < w_o$	$w_e << w_o$	$w_e << w_o$
Number of processes with surface cells	2	4	8	
Imbalance	7	3	1	

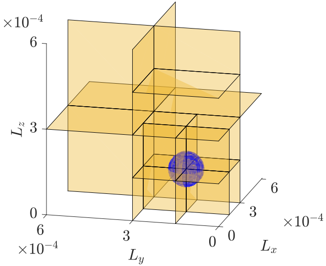
LB 1



LB 2



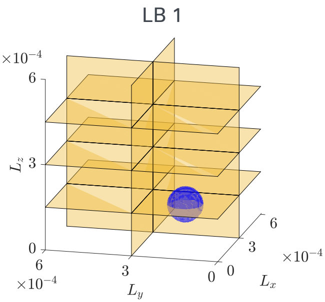
LB 3

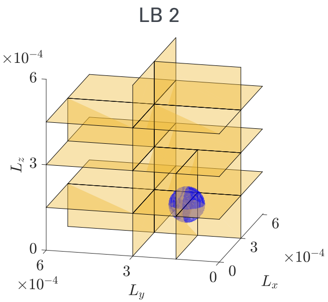


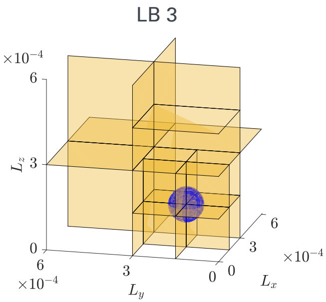
Results

Domain decompositions for Test Case 1

Load Balancing	LB 1	LB 2	LB 3	LB 4
Weight per cell	$w_e = w_o$	$w_e < w_o$	$w_e << w_o$	$w_e << w_o$
Number of processes with surface cells	2	4	8	
Imbalance	7	3	1	**increases**







Results

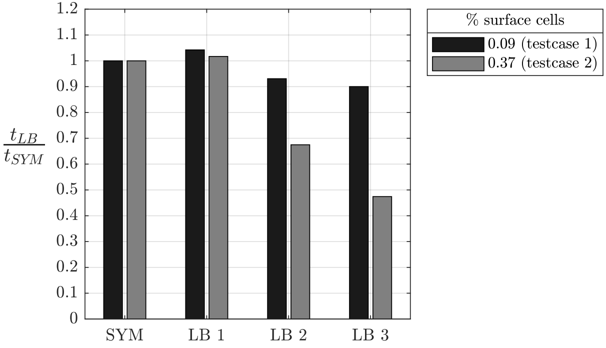
Evaluation of domain decompositions

t_{SYM} : Time per timestep with symmetrical domain decomposition
t_{LB} : Time per timestep with load balanced domain decomposition

Results

Evaluation of domain decompositions

t_{SYM} : Time per timestep with symmetrical domain decomposition
t_{LB} : Time per timestep with load balanced domain decomposition



constant number of cores

Results

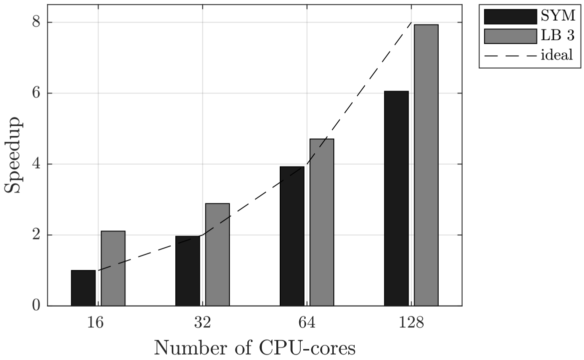
Strong Scaling Test Case 2

$$Speedup = \frac{t_{SYM}}{t_P} \quad (5)$$

Results

Strong Scaling Test Case 2

$$Speedup = \frac{t_{SYM}}{t_P} \tag{5}$$



Conclusions

- Load-Balanced Domain Decomposition
- Nearest neighbor process communication for solidification simulation
- Optimized performance if percentage of boundary cells is high
- Imbalance only in specific Subroutines

Conclusions

- Load-Balanced Domain Decomposition
- Nearest neighbor process communication for solidification simulation
- Optimized performance if percentage of boundary cells is high
- Imbalance only in specific Subroutines

Further work:

- Optimization of domain decomposition
- Investigate optimal cell weight $w_{opt} = f(\text{percentage of boundary cells})$
- Optimization of the complex communication pattern
- Implementation for other phase change processes

University of Stuttgart
Germany

HLRS
High-Performance Computing Center | Stuttgart

Thank you!

Johannes Müller

eMail johannes.mueller@itlr.uni-stuttgart.de

University of Stuttgart
Institute of Aerospace Thermodynamics (ITLR)
Pfaffenwaldring 31
70569 Stuttgart