

MPI and OpenMP in Scientific Software Development

Day 1

Maxim Masterov
SURF, Amsterdam, The Netherlands
09-11.06.2021

Contents

Overview

Day 1

- MPI (recap)
- Connecting to Lisa cluster
- Jacobi solver
- Performance analysis & debugging tools
- Hands-on

Day 2

- **Dr. Gábor Závodsky (UvA)**
- MPI communications
- Domain decomposition
- MPI topologies
- Hands-on

Day 3

- **Dr. Mariano Vázquez (BSC)**
- MPI IO
- Advanced OpenMP
- Hybrid programming
- Hands-on
- **Dr. Satish Kamath (SURF)**

Agenda

Day 1

- MPI (recap)
- Connecting to Lisa cluster
- Jacobi solver
- Hands-on
- Performance analysis & debugging tools
- Hands-on

Disclaimer

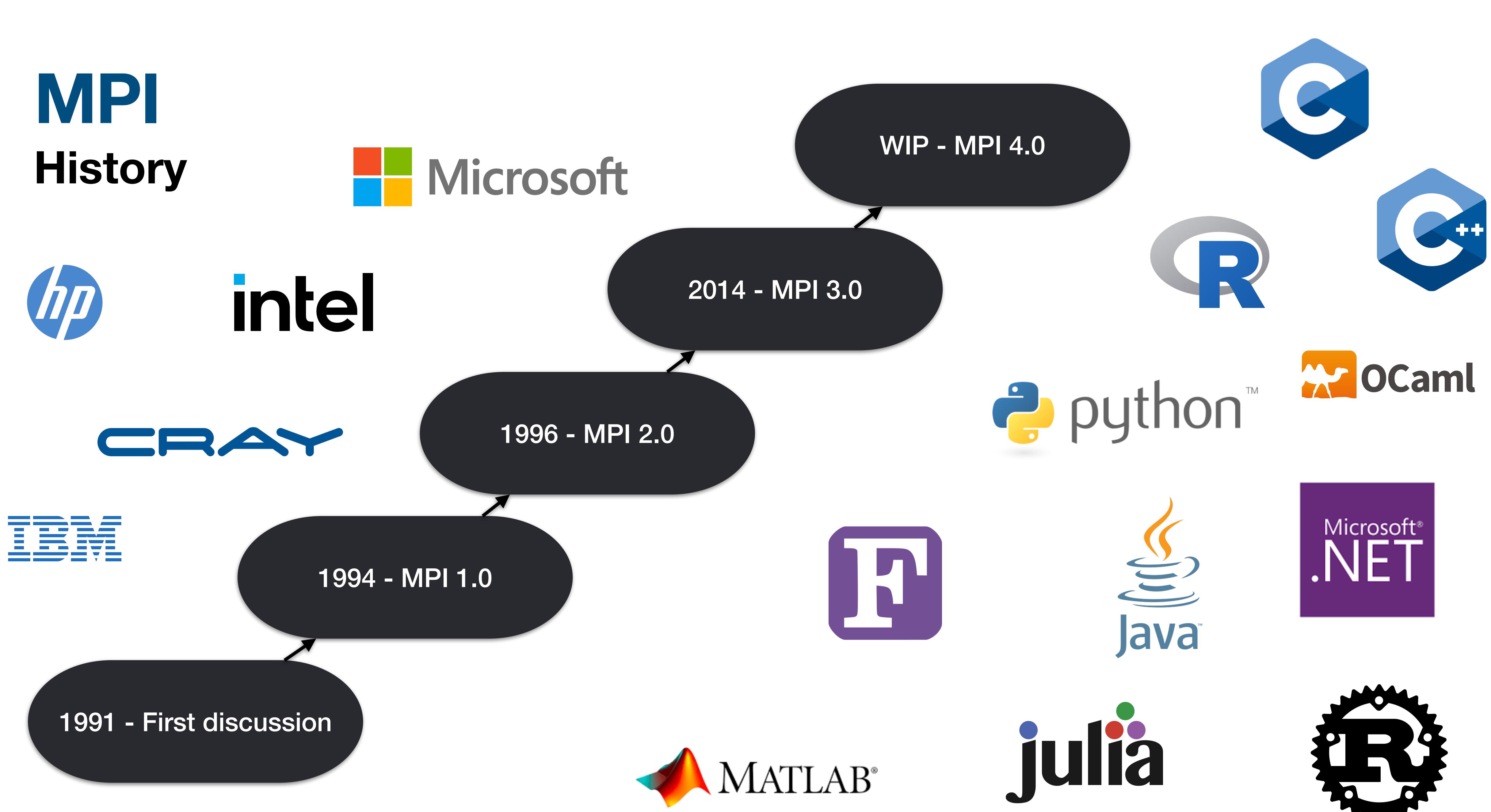
Just a disclaimer

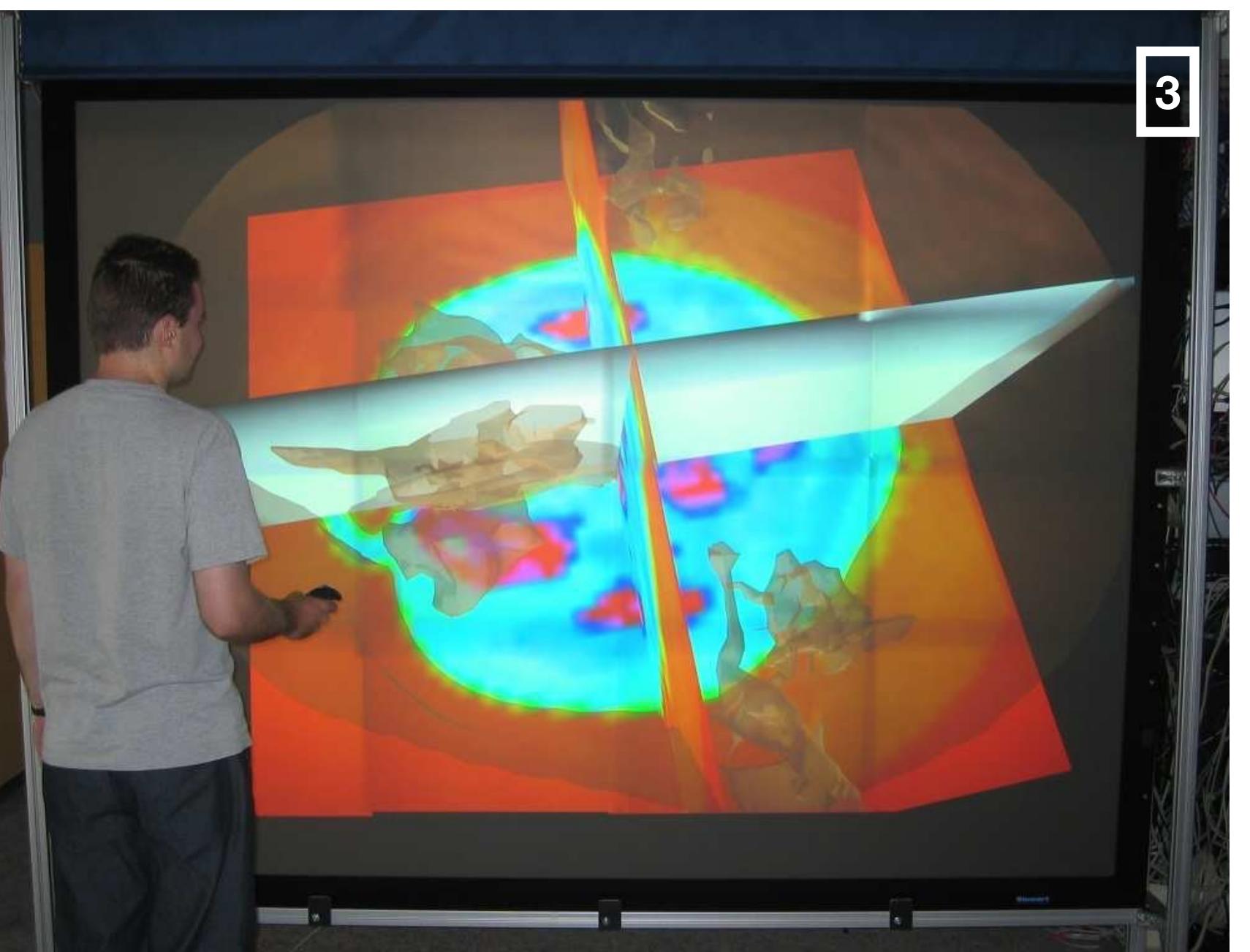
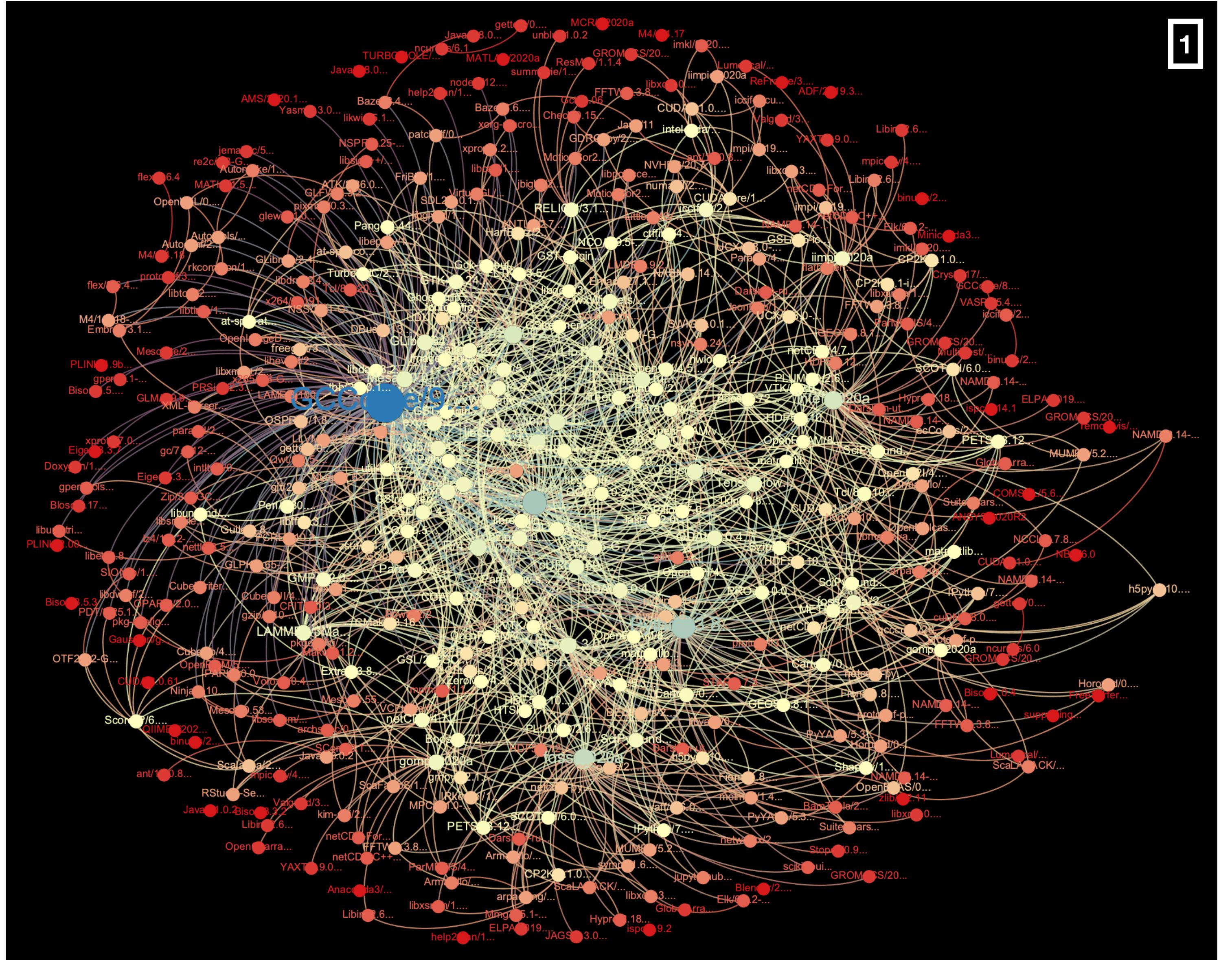
- This is **not the 2nd part** of the course for the “beginners”.
- The whole course is about **hands-on experience**. Please, pay attention, you will write a lot of code.
- All **codes are written in C++**. However, it’s sufficient to know C to do assignments.
- If you are only familiar with **FORTRAN**, please, **check out** some cheat sheets on the Internet.
- All **materials will be available** after the course under MIT license. You can use them or distribute, just keep the references :)
- This is the **1st time** this course is been **offered**. We might have missed something, made typos etc. Your feedback is most appreciated.

Message Passing Interface

MPI

History

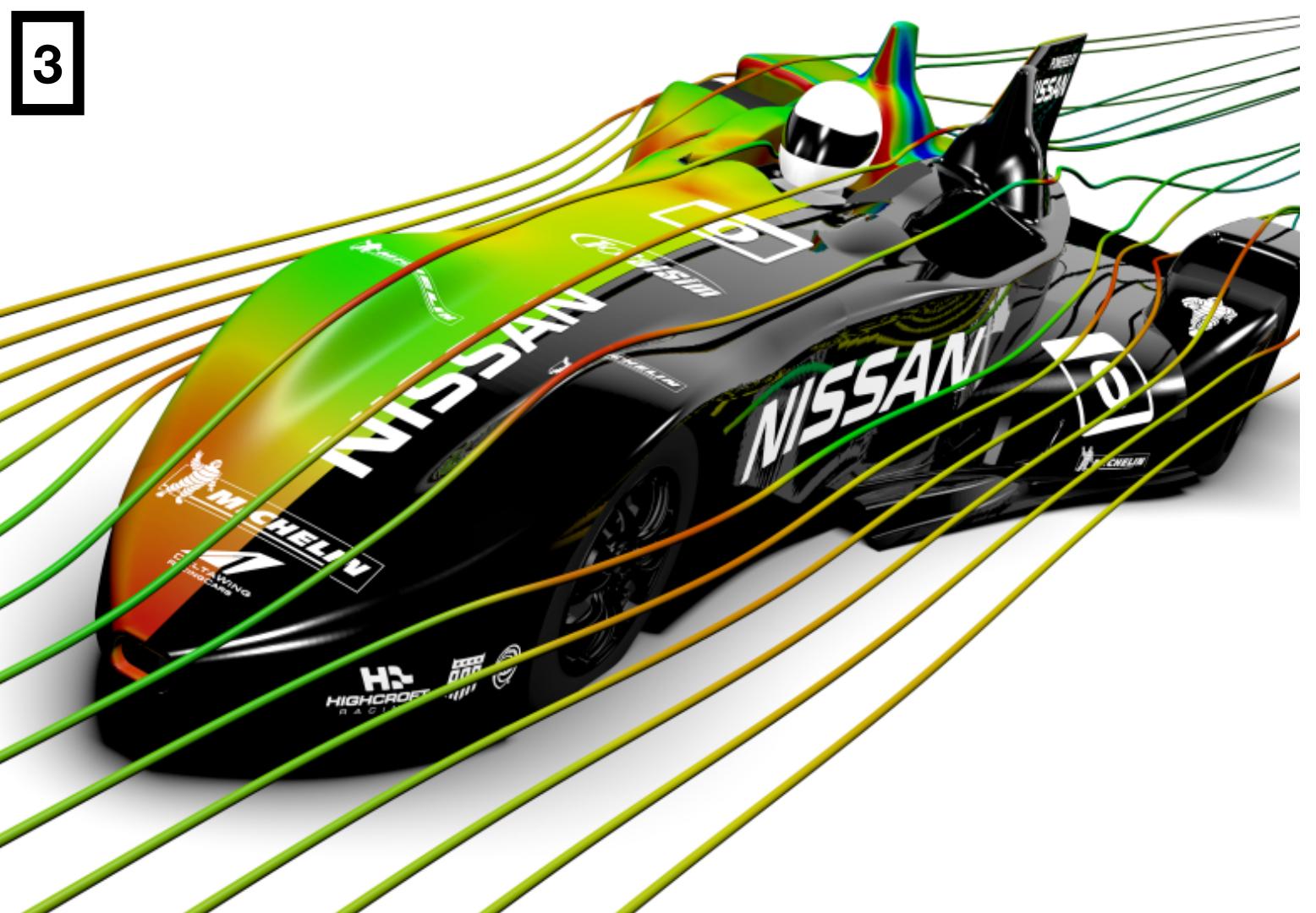
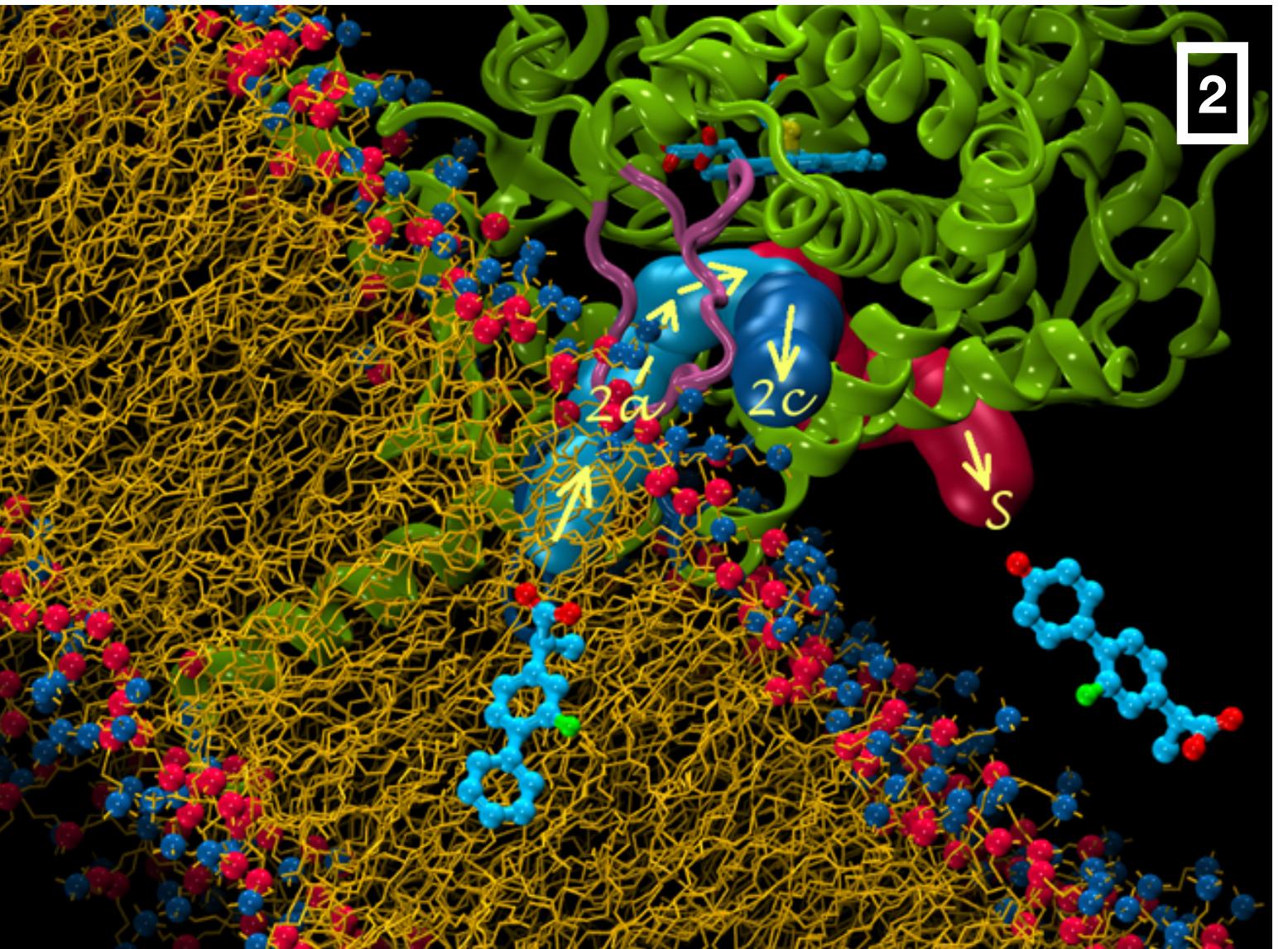
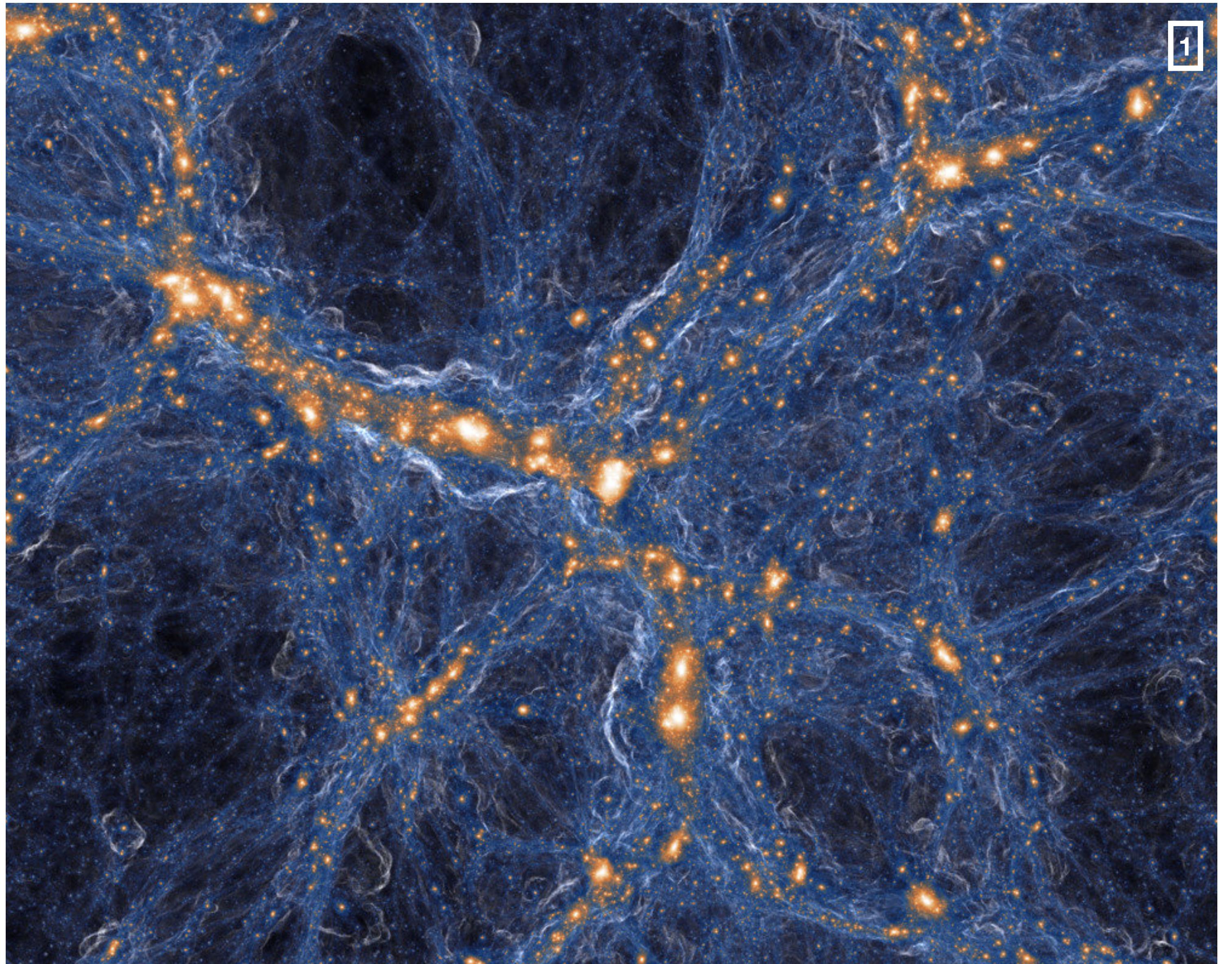




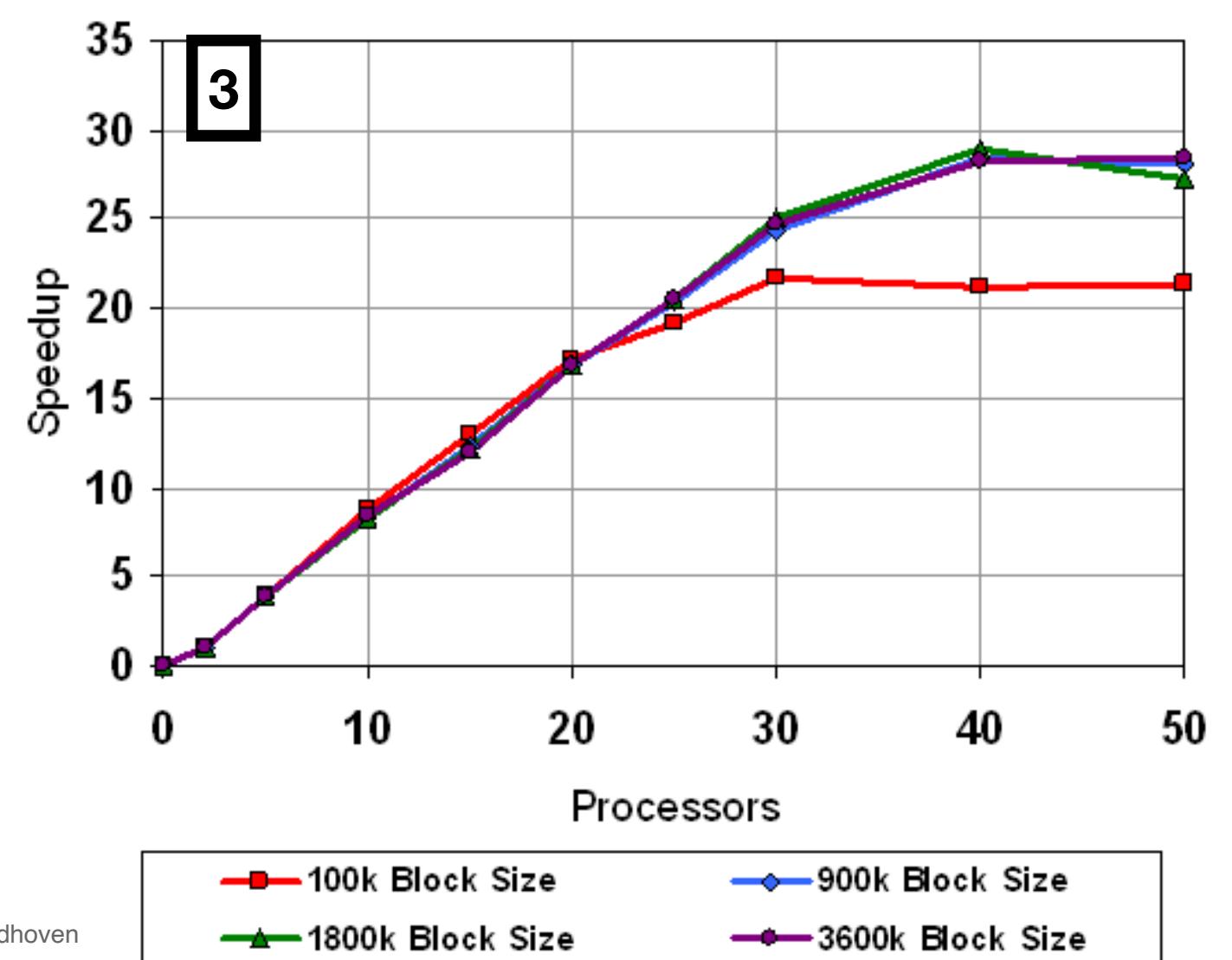
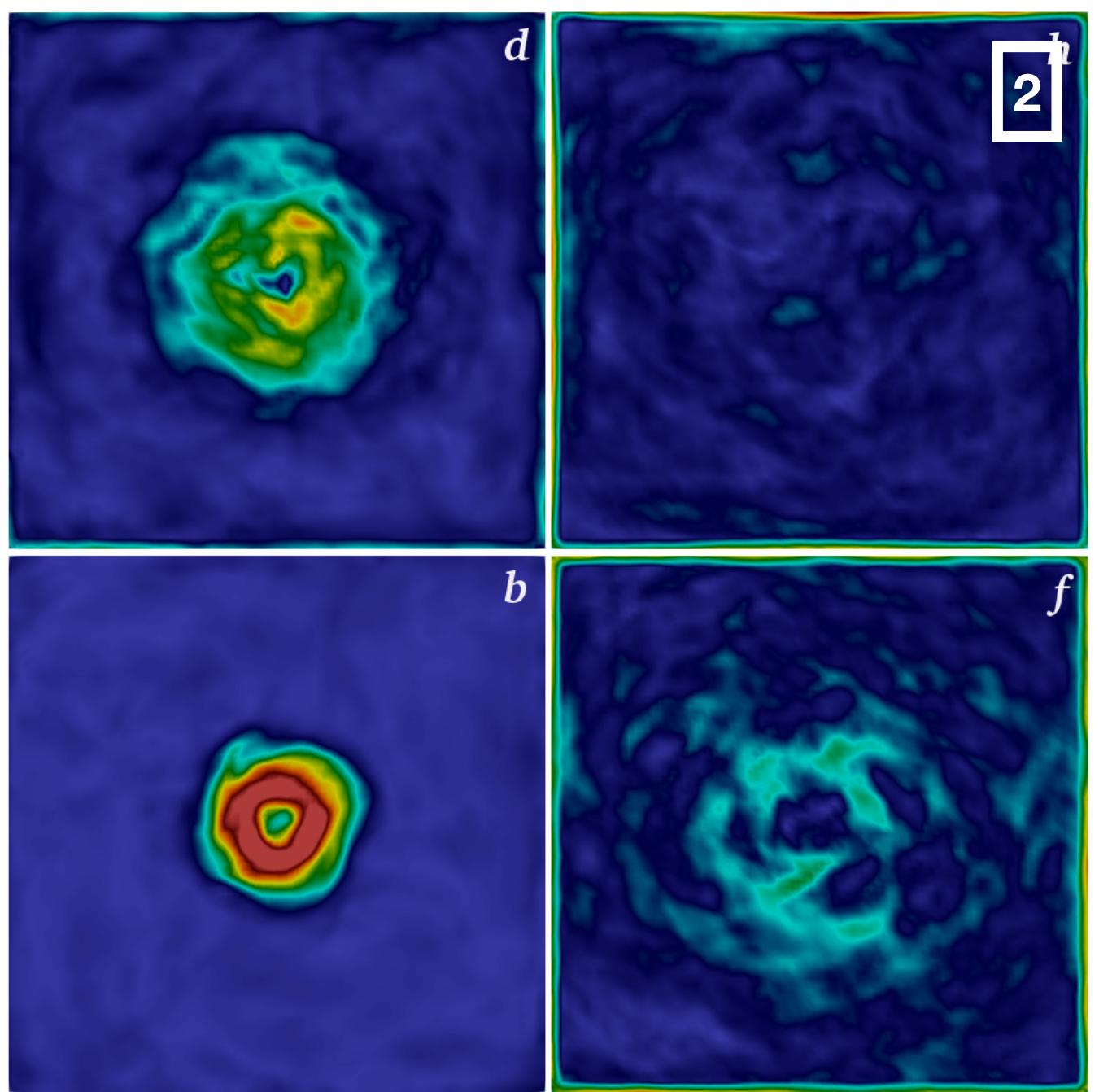
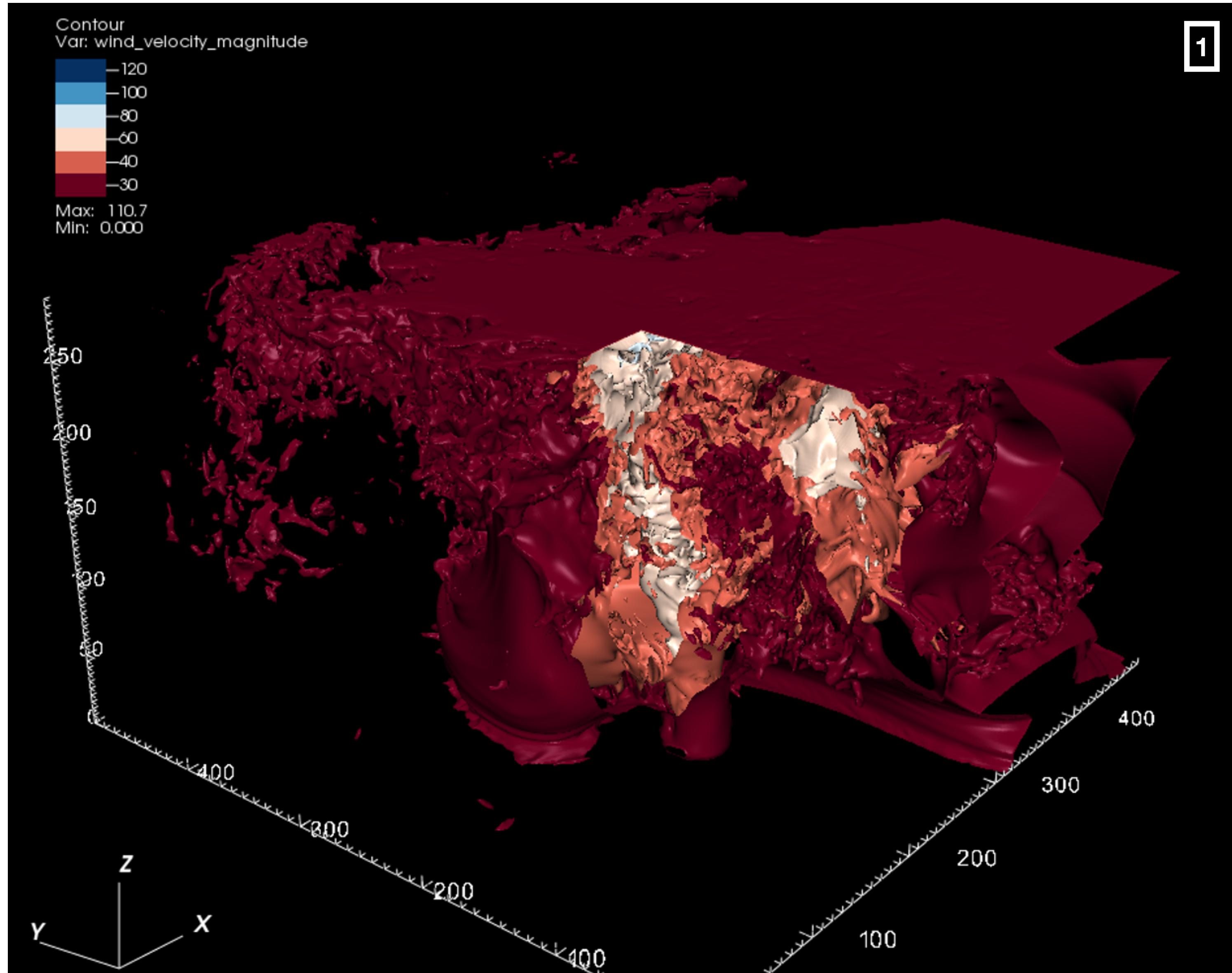
1. Graph of modules installed on Cartesius. Visualised with Gephi.

2. An example of a mobile game. Doolan, Daniel & Tabirca, Sabin. (2007). Bluetooth Gaming with the Mobile Message Passing Interface (MMPI).

³. FlowVR. Visualisation software. Jérémie Allard, Bruno Raffin (2005), A Shader-Based Parallel Rendering Framework, IEEE Visualization 2005 conference proceedings



1. IllustrisTNG. Astrophysics model. <https://phys.org/news/2018-02-astrophysicists-illustristng-advanced-universe-kind.html>
2. VMD. Visualisation software. (2011) *PLoS Computational Biology* Issue Image | Vol. 7(8) August 2011. *PLoS Comput Biol* 7(8)
3. TotalSim. CFD software. <https://www.totalsimulation.co.uk/digital-race-car/>



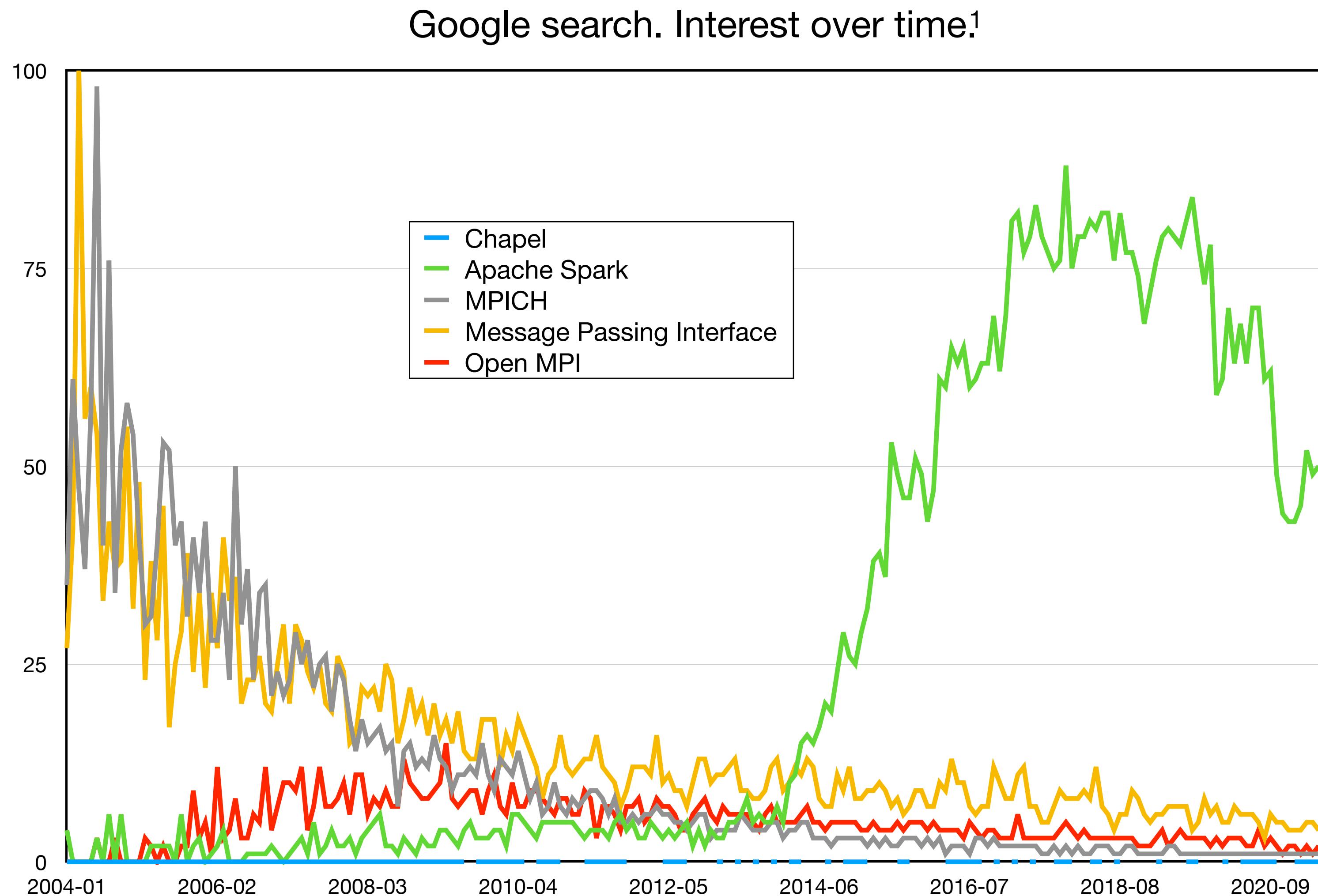
1. VisIt. Visualisation software. <https://www.technologynetworks.com/informatics/articles/visit-application-speeds-visualizations-workloads-empowering-global-research-305352>

2. FoxBerry. Multiphase CFD software. Visualized with ParaView. Masterov, M. V. (2019). Towards industrial-scale bubble columns: the development and application of the high performance computing framework. Technische Universiteit Eindhoven

3. MPIBZIP2. Parallel compressing library. <http://compression.ca/mpibzip2/>

MPI

Alternatives



1. <https://trends.google.com/trends/explore?cat=5&date=all&q=%2Fm%2Fgnckq,%2Fm%2Fndhxqz,%2Fm%2F03z8q2,%2Fm%2F01gb80,%2Fm%2F07yb2g>

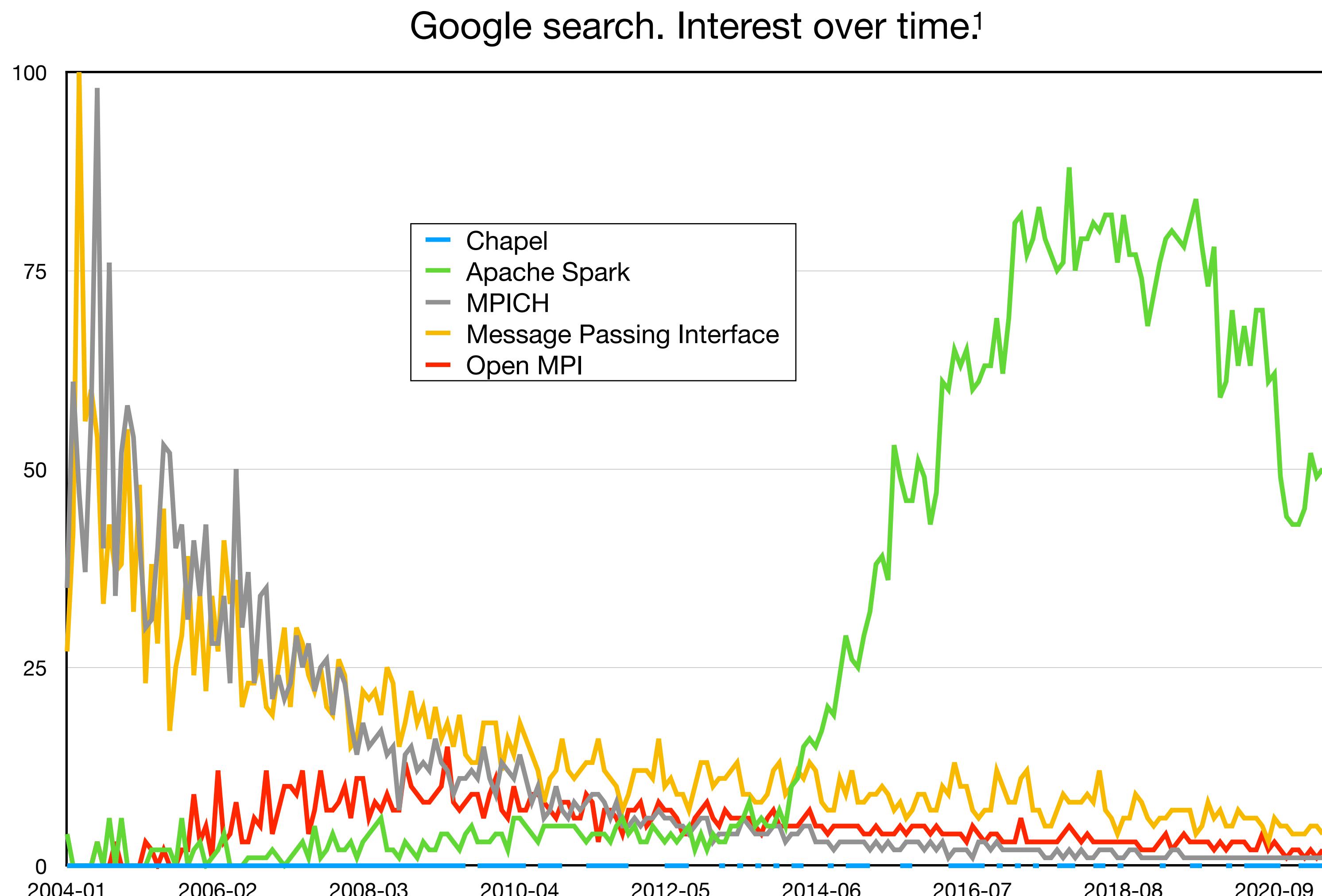
MPI

Alternatives

- New hardware → new software requirements → new standards
- MPI remains the #1 standard for scientific applications, but there are some competitors

Example, 1D diffusion²

Implementation	LoC
MPI+Python	65
Spark+Python	34
Chapel	26



1. <https://trends.google.com/trends/explore?cat=5&date=all&q=%2Fm%2F0gnckq,%2Fm%2F0ndhxqz,%2Fm%2F03z8q2,%2Fm%2F01gb80,%2Fm%2F07yb2g>

2. <https://www.dursi.ca/post/hpc-is-dying-and-mpi-is-killing-it.html#mpi>

MPI Alternatives

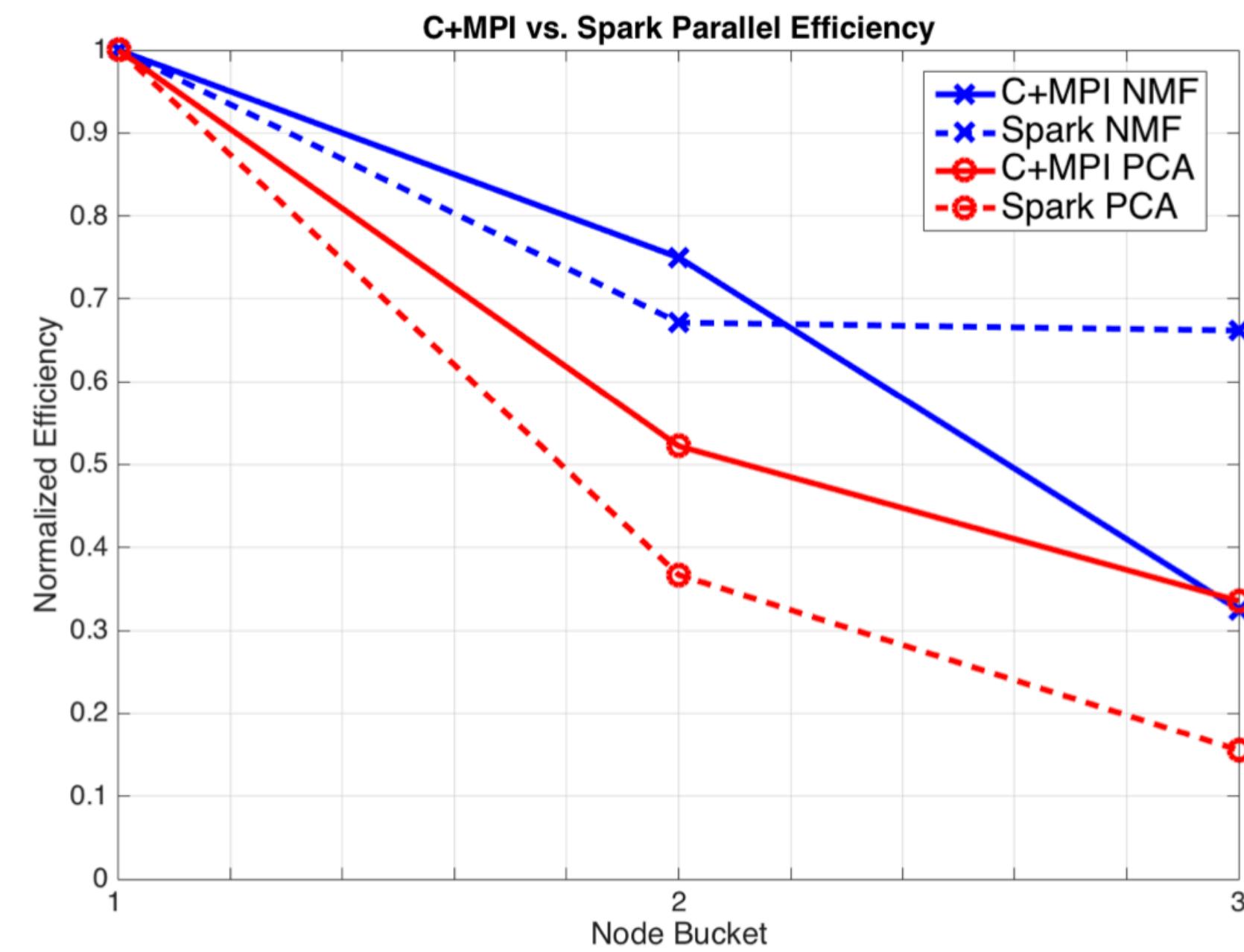


Figure 4: Comparison of parallel efficiency for C+MPI and Spark. The x-axis label “Node Bucket” refers to the node counts. For NMF these are 50, 100, and 300 nodes (left to right) and 100, 300, and 500 nodes for PCA. For both algorithms, efficiency is measured relatively to the performance at the smallest node count.

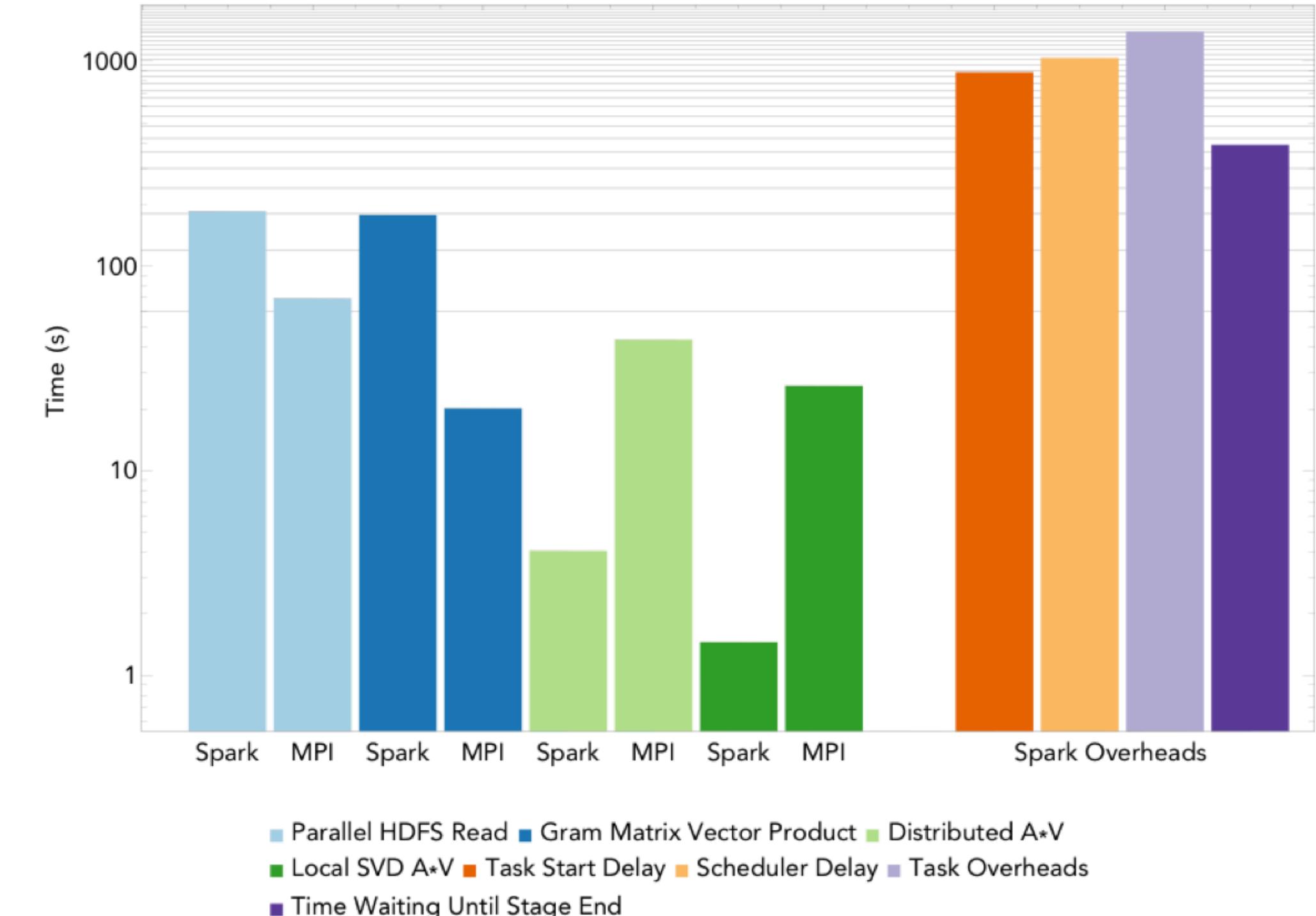
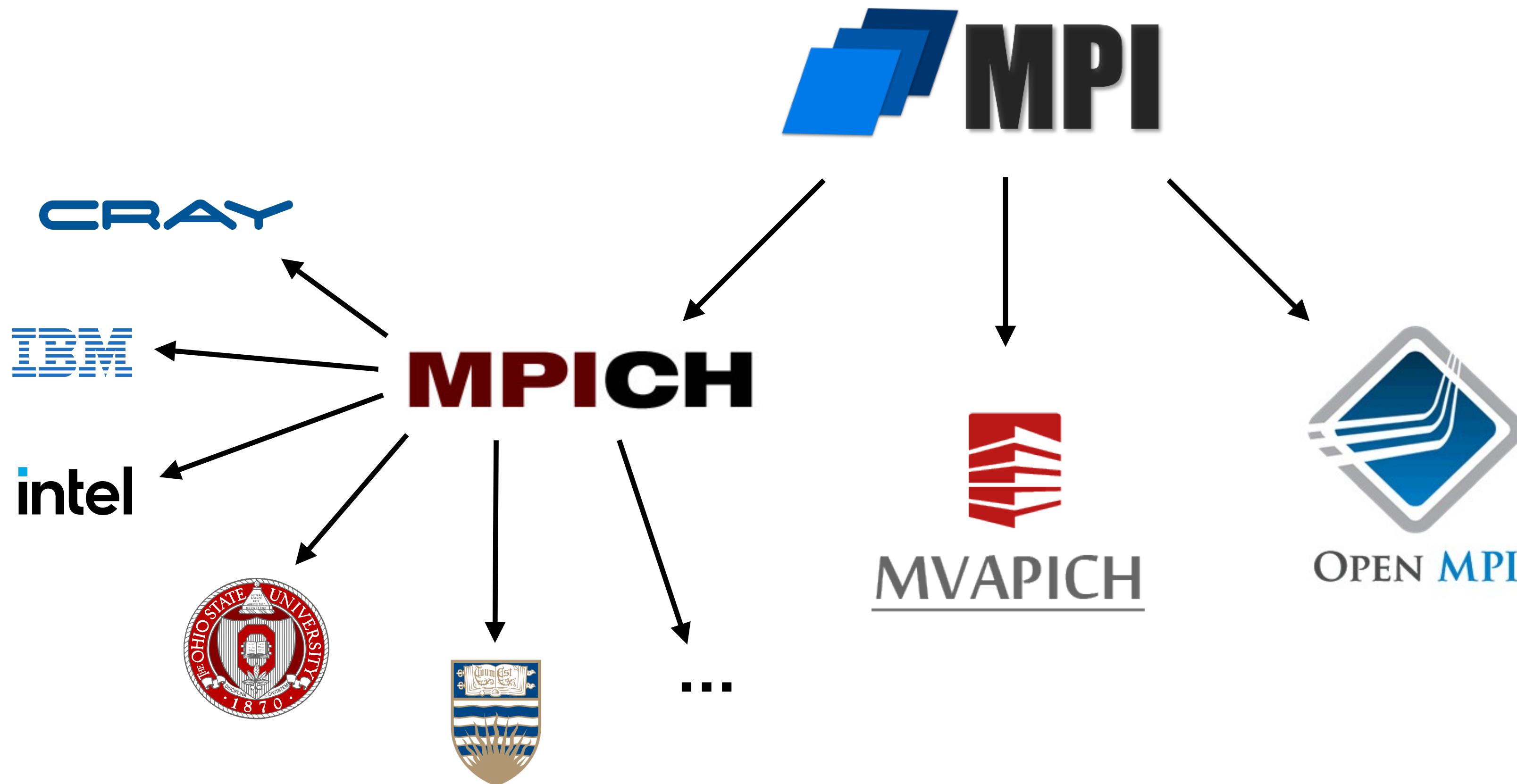


Figure 6: Running time comparison of the Spark and MPI implementations of PCA on the 16TB Atmosphere matrix. Each bin depicts the sum, over all stage, of the time spent in that bin by the average task within a stage.

MPI

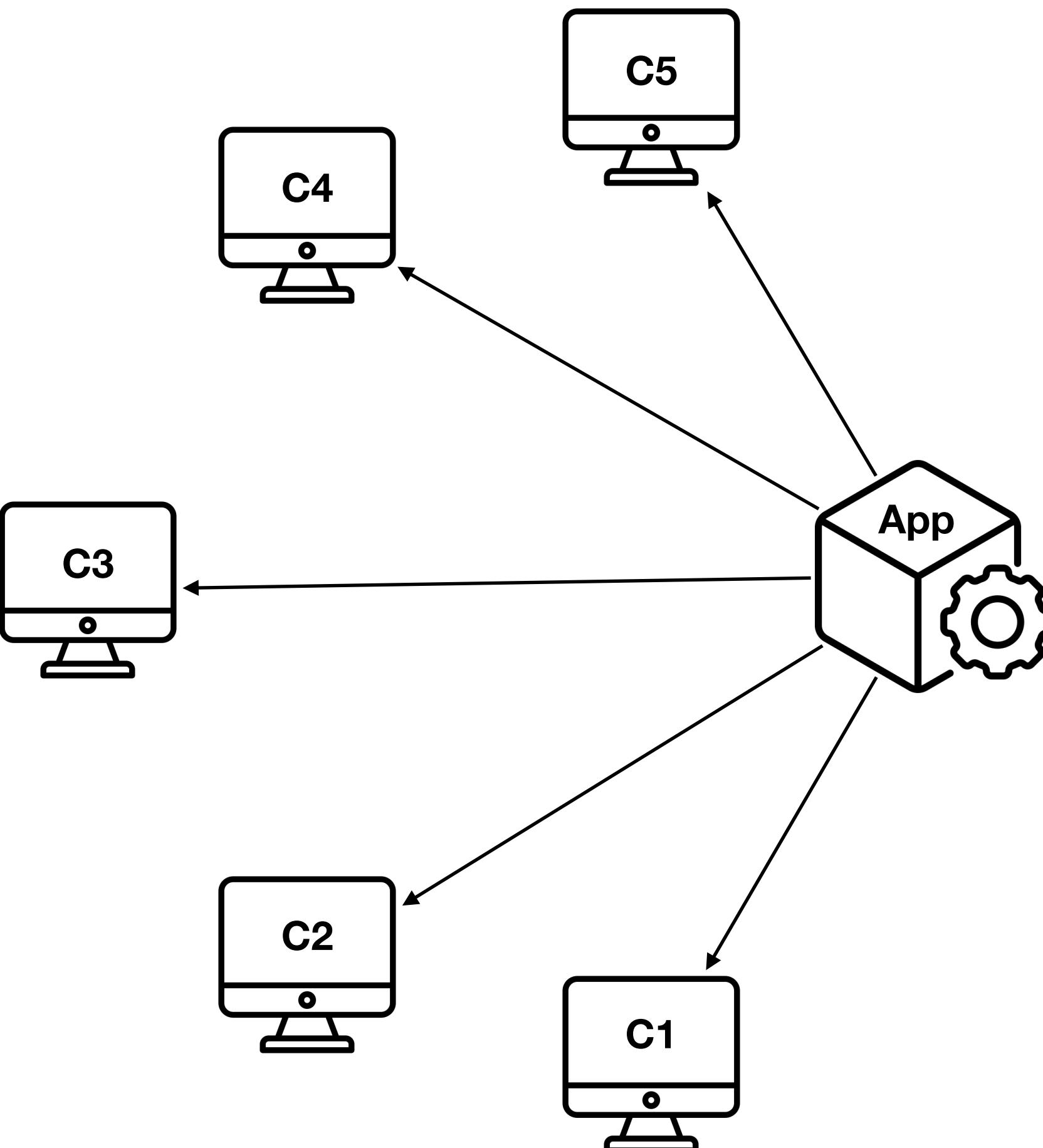
Versions



MPI

Idea

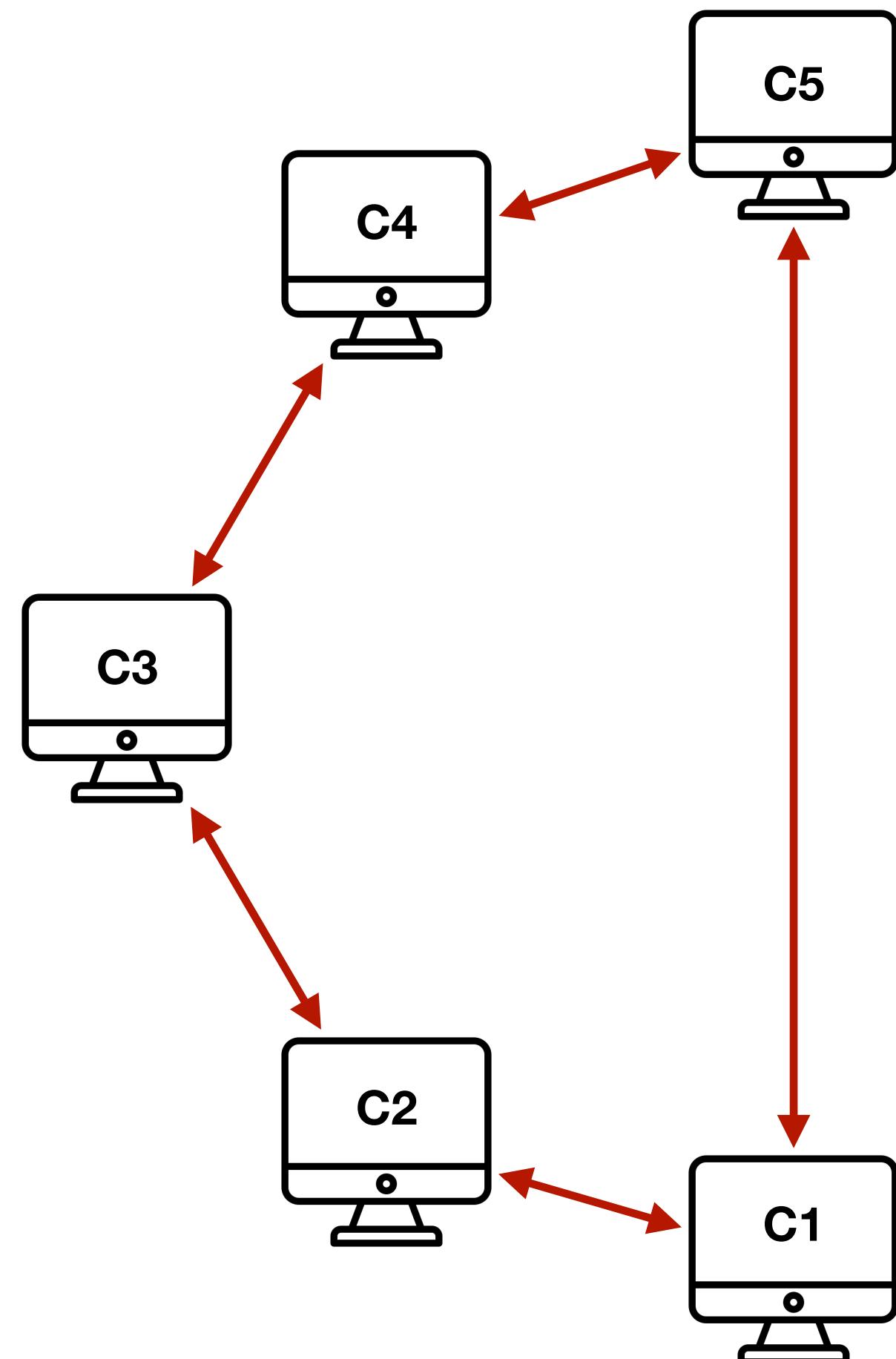
- Distribution of the workload across multiple processes



MPI

Idea

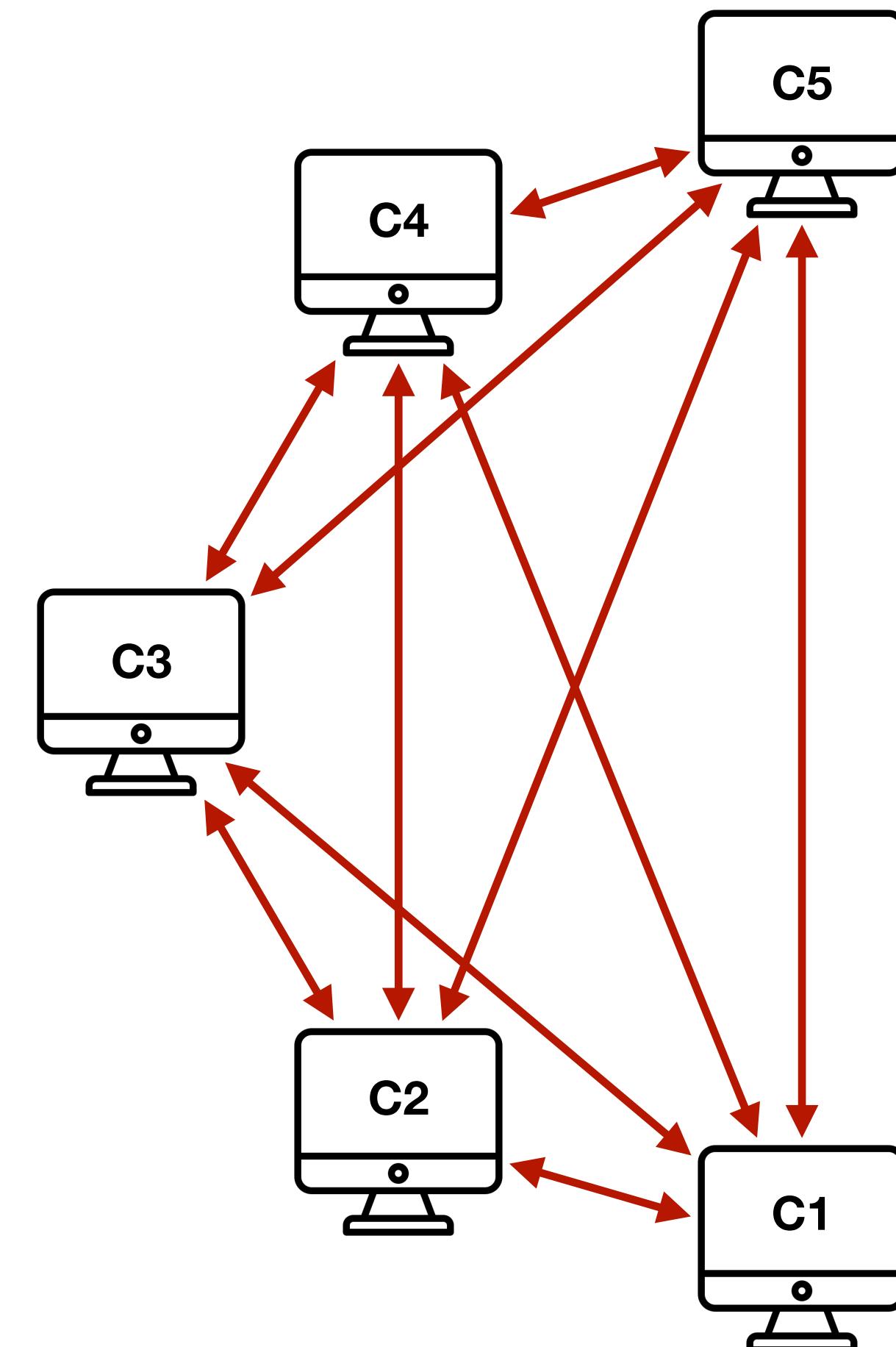
- Distribution of the workload across multiple processes
- Main influence from:
 - the network



MPI

Idea

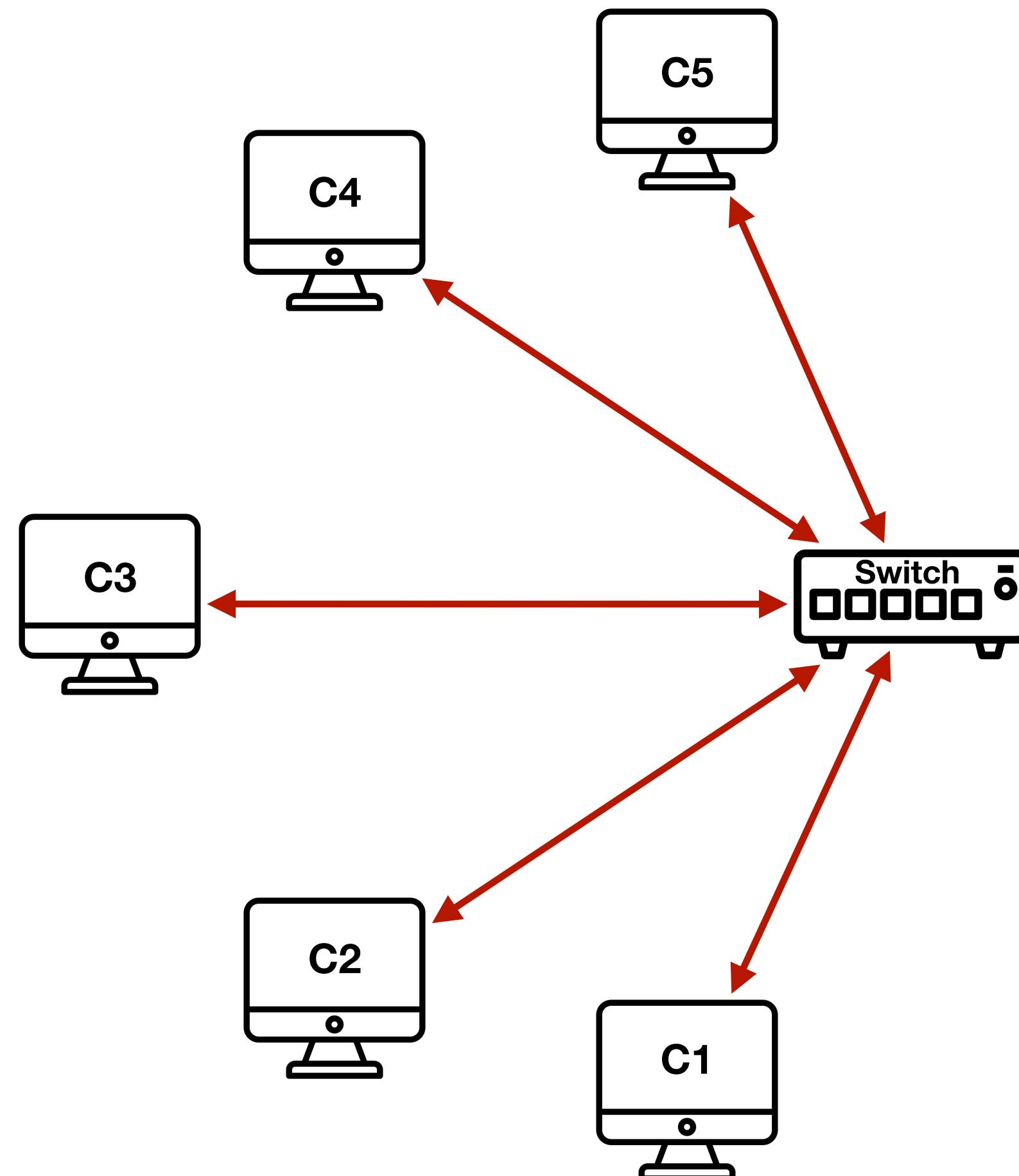
- Distribution of the workload across multiple processes
- Main influence from:
 - the network



MPI

Idea

- Distribution of the workload across multiple processes
- Main influence from:
 - the network

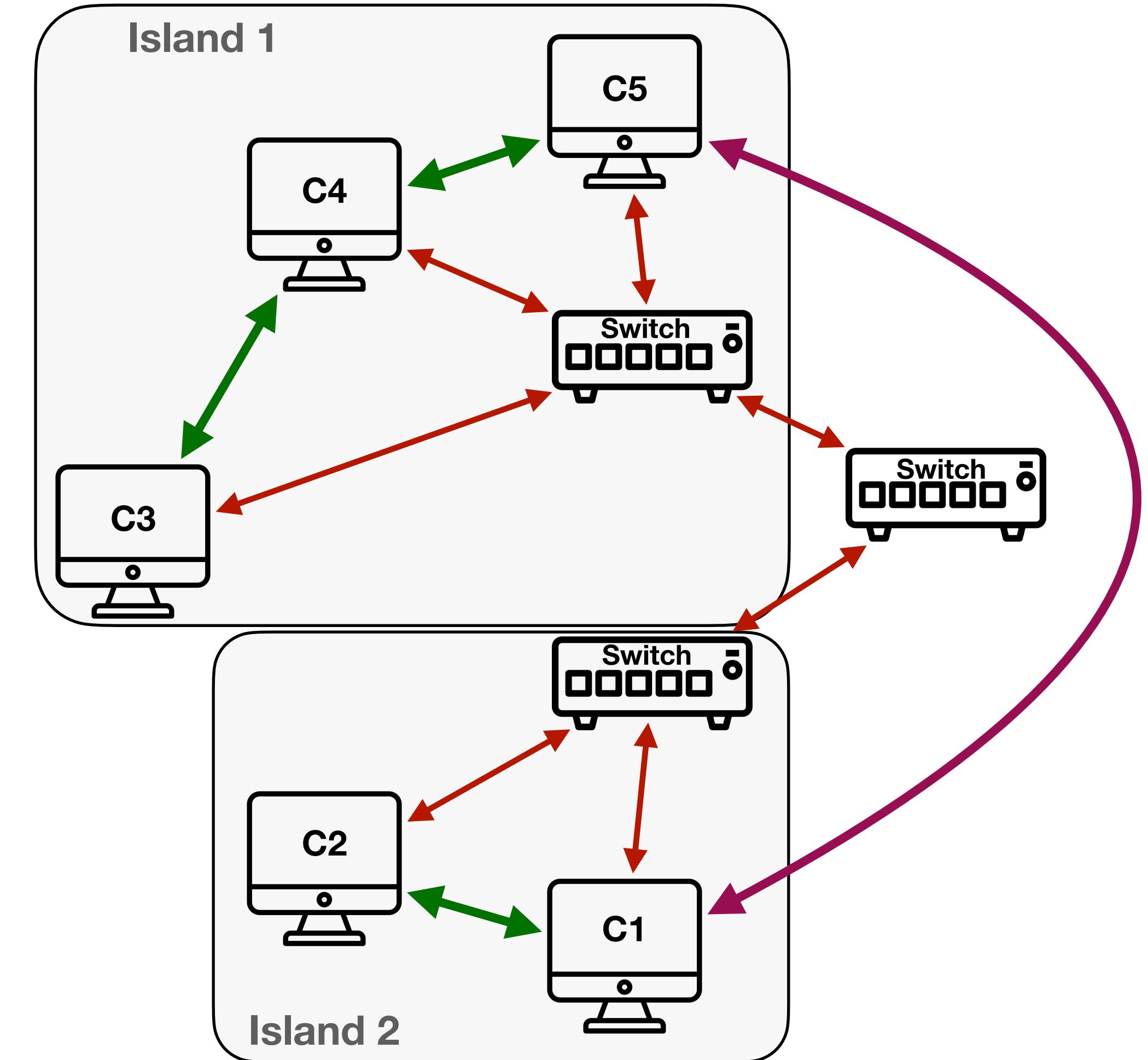


MPI

Idea

- Distribution of the workload across multiple processes
- Main influence from:
 - the network
 - the communication frequency
 - the message size
 - the file system

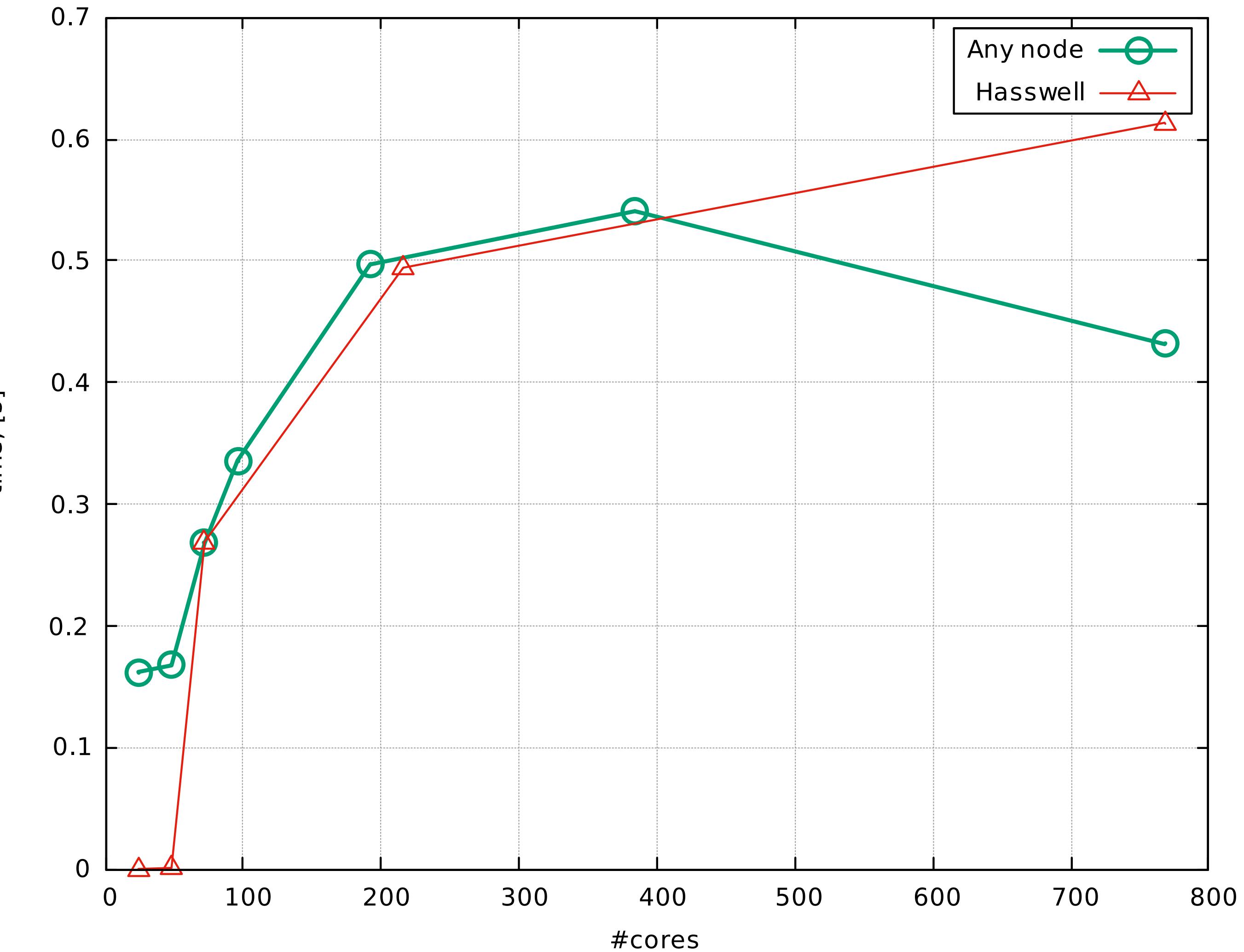
Will come back to it later



MPI

Message size

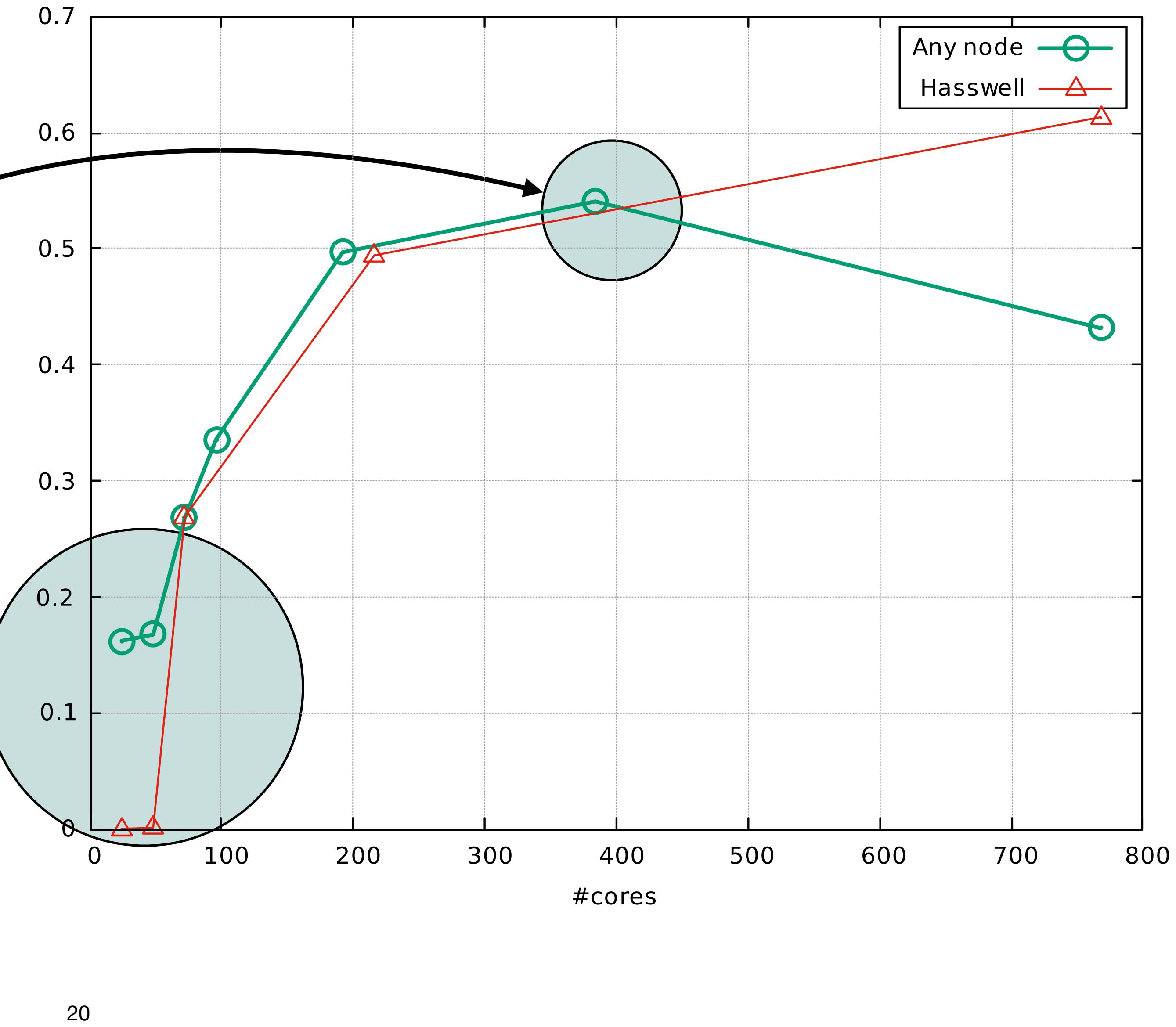
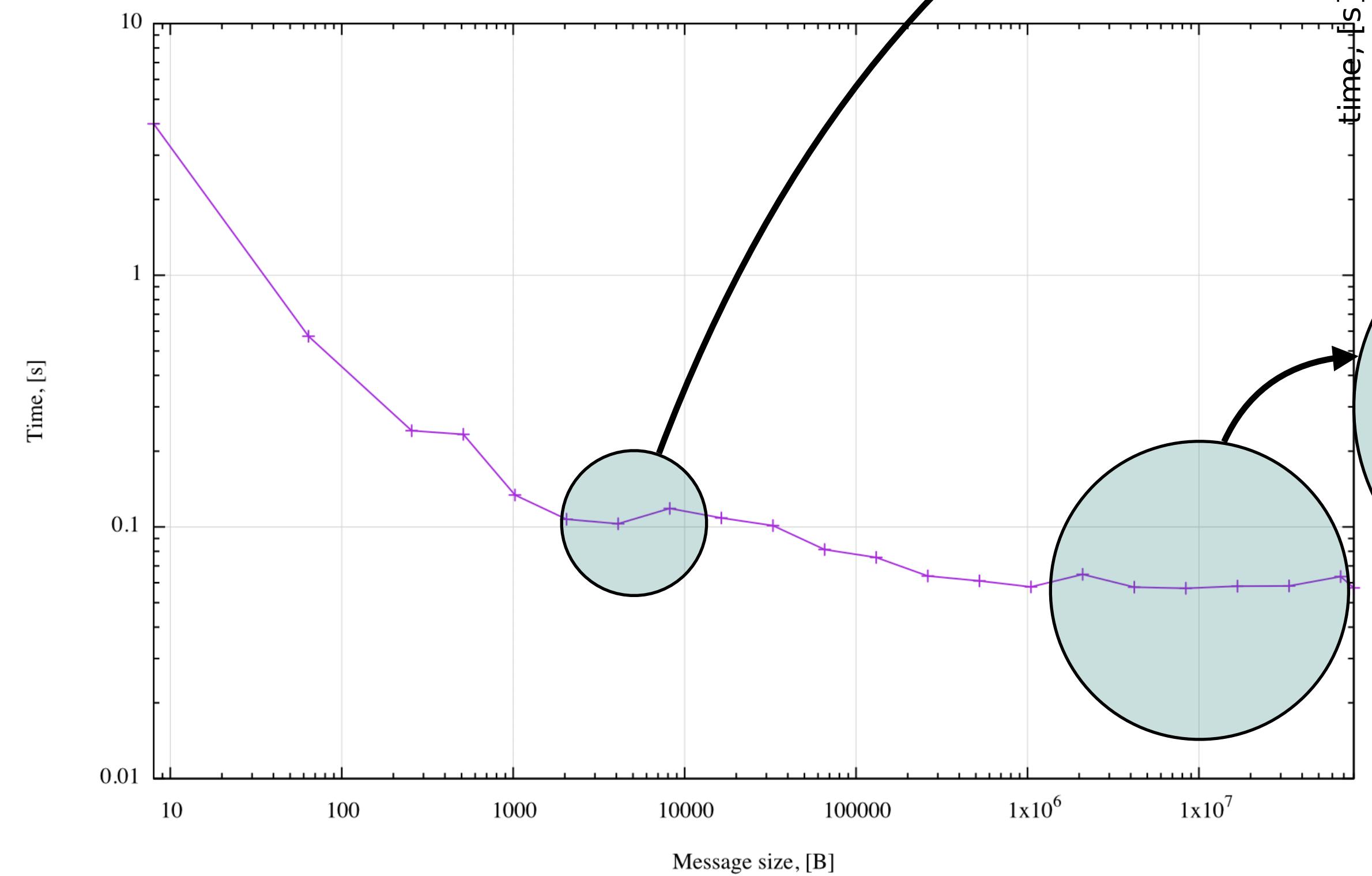
- Communication on Cartesius
- CFD code, FoxBerry
- Structured grid with 27M DoFs
- Double precision FP
- Non-blocking communication
- Buffered messaging
- The message size at 700 nodes is ~3KB



MPI

Message size

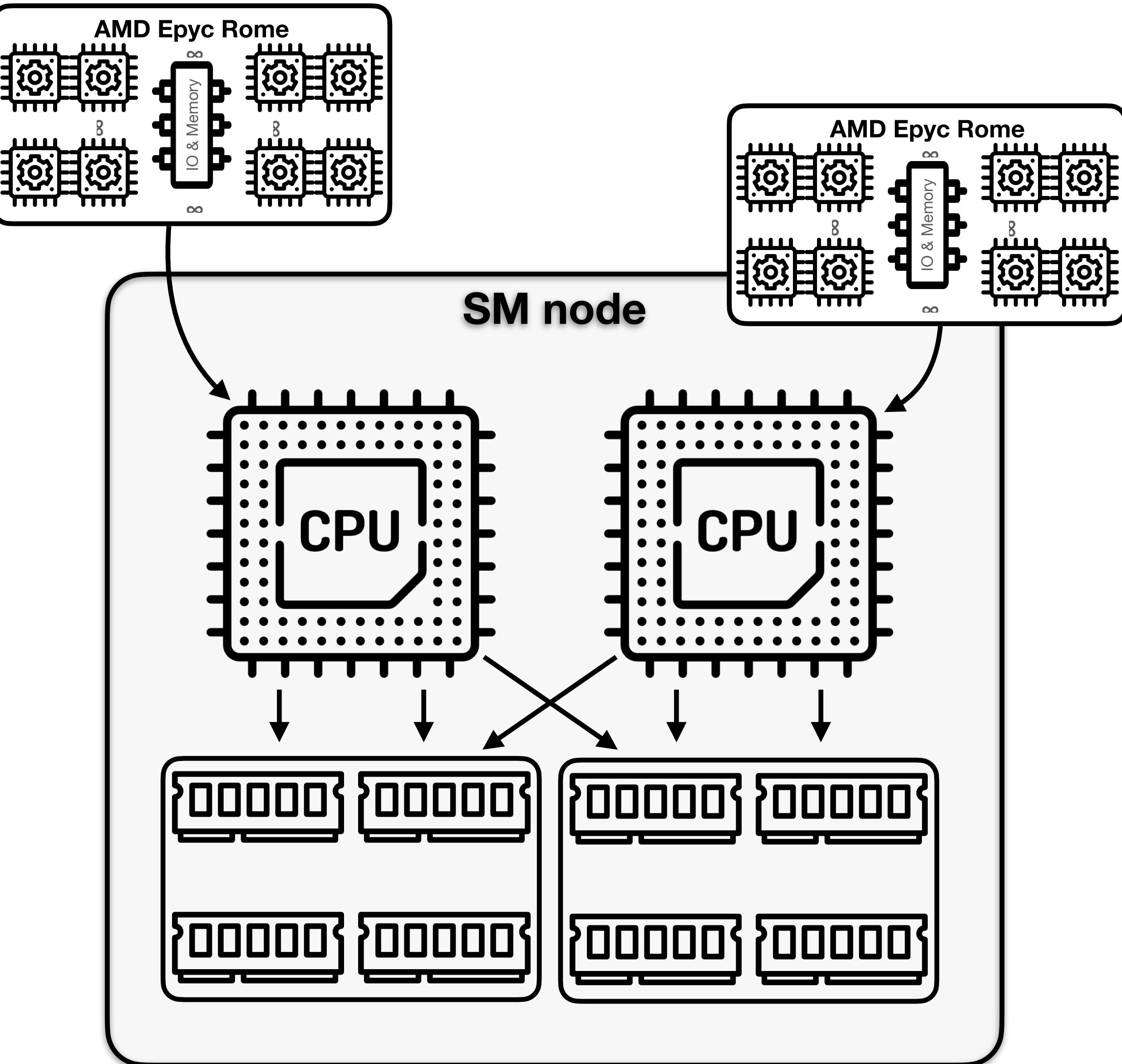
- Communication on Cartesius
- CFD code, FoxBerry



MPI

Shared Memory Nodes

- Behaviour on SM nodes is implementation-specific and is not described by the standard
- Some MPI libraries avoid “standard” communications on SM nodes thus, improving the performance (see BTL in OpenMPI)
- Some implementations provide run-time tuning parameters for SM execution
- There is no such problem as memory affinity (as with threads)



1. https://www.flaticon.com/free-icon/ram_900330

2. https://www.flaticon.com/free-icon/cpu_689379

3. https://www.flaticon.com/free-icon/cpu_4272570

4. https://www.flaticon.com/free-icon/ram_3076285

MPI

Rules of thumb

- Follow the **KISS** principle - keep it simple stupid
- Make it **maintainable**
- Make it **efficient**
- Unless it is completely necessary:
 - Avoid complex data structures
 - Avoid complex logic
- Always do **refactoring**

```
#include <cmath>
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
    // Initialize the MPI environment
    MPI_Init(&argc, &argv);

    // Get the number of processes
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    // Get the rank of the process
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

    // Get the name of the processor
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);

    // Print off a hello world message
    printf("Hello world from processor %s, "
           "rank %d out of %d processors\n",
           processor_name, world_rank, world_size);

    // Finalize the MPI environment.
    MPI_Finalize();
}
```

MPI

Rules of thumb

- Follow the **KISS** principle - keep it simple stupid
- Make it **maintainable**
- Make it **efficient**
- Unless it is completely necessary:
 - Avoid complex data structures
 - Avoid complex logic
- Always do **refactoring**

Example of a real code

```
...  
    if(greater1 && lesser1)  
    {  
        nozzle_1_send_to_root_proc = true;  
        for(size_t i = 0; i < (size_t)(max_graph_neighbours); ++i)  
        {  
            ++comm_look_up_table(my_rank, source[i]);  
            MPI_Isend(&nozzle_1_send_to_root_proc,  
                      1,  
                      MPI_CXX_BOOL,  
                      source[i],  
                      nozzle1_tag1,  
                      _grid->GetLocMPIComm(),  
                      &nozzle1_snd_rqst[i]);  
        }  
    }  
  
    MPI_Bcast(comm_look_up_table.data(),  
              comm_look_up_table.size(),  
              MPI_INT,  
              rank_with_nozzle1_center,  
              internal_MPI_Comm);  
  
    MPI_Waitall(max_graph_neighbours, nozzle1_snd_rqst, nozzle1_status);  
  
    MPI_Request rqst;  
    int count_to_root_proc = 0;  
    if(comm_look_up_table(rank_with_nozzle1_center, my_rank))  
    {  
        int dummy = 0, weighted = 0;  
        MPI_Recv(&nozzle_1_send_to_root_proc,  
                 1,  
                 MPI_CXX_BOOL,  
                 rank_with_nozzle1_center,  
                 nozzle1_tag1,  
                 _grid->GetLocMPIComm(),  
                 &nozzle1_status_point1);  
  
        MPI_Dist_graph_neighbors_count(_grid->GetLocMPIComm(),  
                                       &max_graph_neighbours, &dummy, &weighted);  
        source.resize(max_graph_neighbours);  
        source_weights.resize(max_graph_neighbours);  
        dest.resize(max_graph_neighbours);  
        dest_weights.resize(max_graph_neighbours);  
        MPI_Dist_graph_neighbors(_grid->GetLocMPIComm(),  
                               max_graph_neighbours,  
                               source.data(),  
                               source_weights.data(),  
                               max_graph_neighbours,  
                               dest.data(),  
                               dest_weights.data());  
    }  
}
```

MPI

MPI code

```
int MPI_Send(const void *buf,  
            int count,  
            MPI_Datatype datatype,  
            int dest,  
            int tag, MPI_Comm comm);  
  
int MPI_Recv(void *buf,  
            int count,  
            MPI_Datatype datatype,  
            int source,  
            int tag, MPI_Comm comm,  
            MPI_Status *status);
```

```
class MPIManager{  
public:  
  
    inline int send(const int* buff, int size, int pid,  
    int tag)  
    {  
        MPI_Send(  
            &buff,  
            size,  
            MPI_INT,  
            pid,  
            tag,  
            mpi_comm);  
        // handle errors  
    }  
  
    inline int recv(int* buff, int size, int pid, int  
tag)  
    {  
        MPI_Status status;  
        MPI_Recv(  
            &buff,  
            size,  
            MPI_INT,  
            pid,  
            tag,  
            mpi_comm,  
            &status);  
        // handle errors  
    }  
  
private:  
    MPI_Comm mpi_comm;  
};
```

MPI

MPI code

```
int main(int argc, char** argv) {
    // Initialize the MPI environment
    MPI_Init(&argc, &argv);

    // Get the number of processes
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    // Get the rank of the process
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

    // Get the name of the processor
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);

    // Print off a hello world message
    printf("Hello world from processor %s, "
        "rank %d out of %d processors\n",
        processor_name, world_rank, world_size);

    // Finalize the MPI environment.
    MPI_Finalize();
}
```

```
#include <cmath>
#include <mpi.h>
#include <stdio.h>

class MPIManager {
    MPI_Comm comm;
public:
    MPIManager() : comm(MPI_COMM_WORLD) { }
    void initialize(int argc, char** argv) { ... }
    void finalize() { ... }
    char *getProcName() { ... }
    int getMyRank() { ... }
    intgetNumProcs() { ... }
};

int main(int argc, char** argv) {
    MPIManager mpi_manager;

    // Initialize the MPI environment
    mpi_manager.initialize(argc, argv);

    // Print off a hello world message
    printf("Hello world from processor %s, "
        "rank %d out of %d processes\n",
        mpi_manager.getProcName(),
        mpi_manager.getMyRank(),
        mpi_manager.getNumProcs());

    // Finalize the MPI environment.
    mpi_manager.finalize();
}
```

Connecting to Lisa

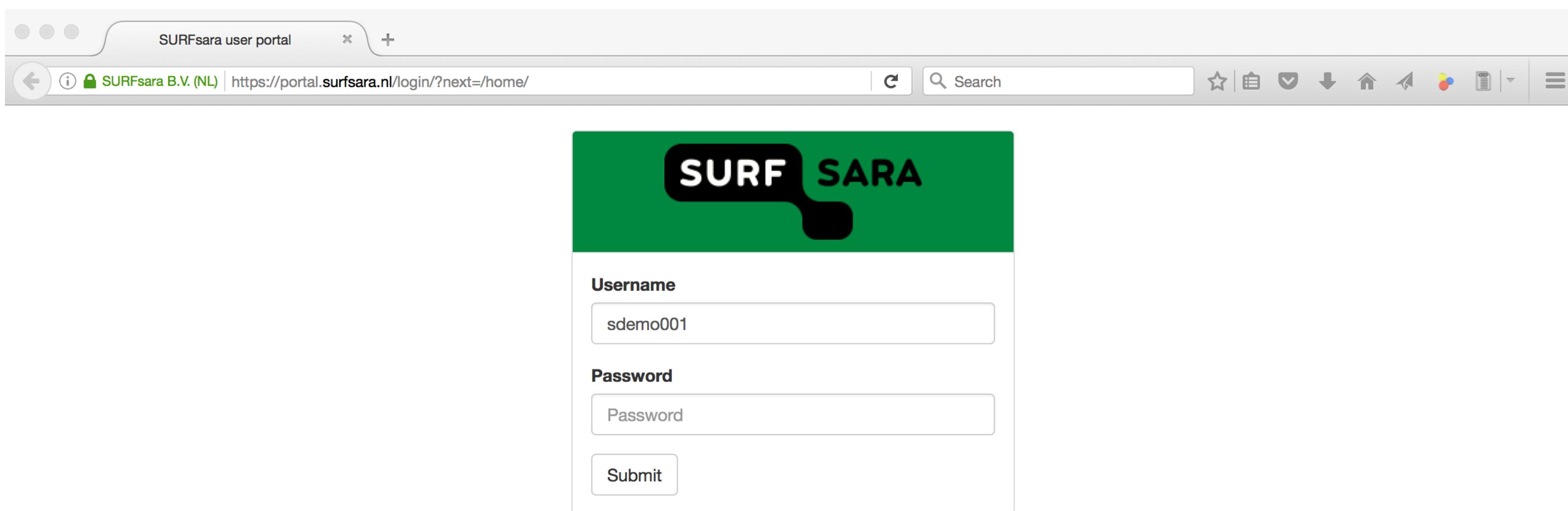
Lisa

Connecting to

- **Windows OS:**
 - MobaXterm: <https://mobaxterm.mobatek.net/>
(download portable version)
 - Putty
- **macOS and Linux OS:**
 - You are already well equipped
 - Domain name: **lisa.surfsara.nl**
 - Use **-X** or **-Y** flag in the **ssh** command to allow **X11 forwarding** (for visualisation)
 - macOS users should install XQuartz

Lisa

Connecting to



<https://portal.surfsara.nl>

Lisa

Connecting to

The screenshot shows the SURFsara user portal interface. At the top, there is a logo for "SURF SARA". Below it, a welcome message says "Welcome, you are currently logged in as Carlos Teijeiro (uid: carlost)". On the left, a sidebar menu includes "Home" (selected), "Your Profile", "Accounting", "Public ssh keys", "Change password", "Helpdesk", and "Logout". The main content area has a title "SURFsara user portal" and a sub-section "Welcome to the SURFsara user Portal" which states: "SURFsara supports researchers in the Netherlands and works closely together with the academic community and industry." It also lists what the portal allows: "View your login profile", "Check your accounting details (if applicable)", "Change your password", and "Contact the helpdesk". To the right, there is a "Current system status" section showing a list of services with their status icons:

System	Status
Cartesius	✓
Lisa	⚠
Data Archive	✓
EPIC PID's	✓
ResearchDrive	✓
B2SAFE	✓
Grid	✓
HPC Cloud	⚠
Hathi Hadoop	⚠

Lisa

Connecting to

The screenshot shows the SURFsara user portal interface. At the top, there is a logo for "SURF SARA". Below it, a welcome message says "Welcome, you are currently logged in as Carlos Teijeiro (uid: carlost)". On the left, a sidebar menu includes "Home" (selected), "Your Profile", "Accounting", "Public ssh keys", "Change password" (highlighted with a red oval), "Helpdesk", and "Logout". The main content area has a title "SURFsara user portal" and a "Welcome to the SURFsara user Portal" message. It also lists the "SURFsara supports researchers in the Netherlands and works closely together with the academic community and industry." Below this, a list of what the portal allows includes: "View your login profile", "Check your accounting details (if applicable)", "Change your password", and "Contact the helpdesk". To the right, a "Current system status" section displays a list of systems with their status icons: Cartesius (green checkmark), Lisa (orange warning triangle), Data Archive (green checkmark), EPIC PID's (green checkmark), ResearchDrive (green checkmark), B2SAFE (green checkmark), Grid (green checkmark), HPC Cloud (orange warning triangle), and Hathi Hadoop (green checkmark).

Welcome, you are currently logged in as Carlos Teijeiro (uid: carlost)

SURFsara user portal

Welcome to the SURFsara user Portal

SURFsara supports researchers in the Netherlands and works closely together with the academic community and industry.

The SURFsara user portal allows you to:

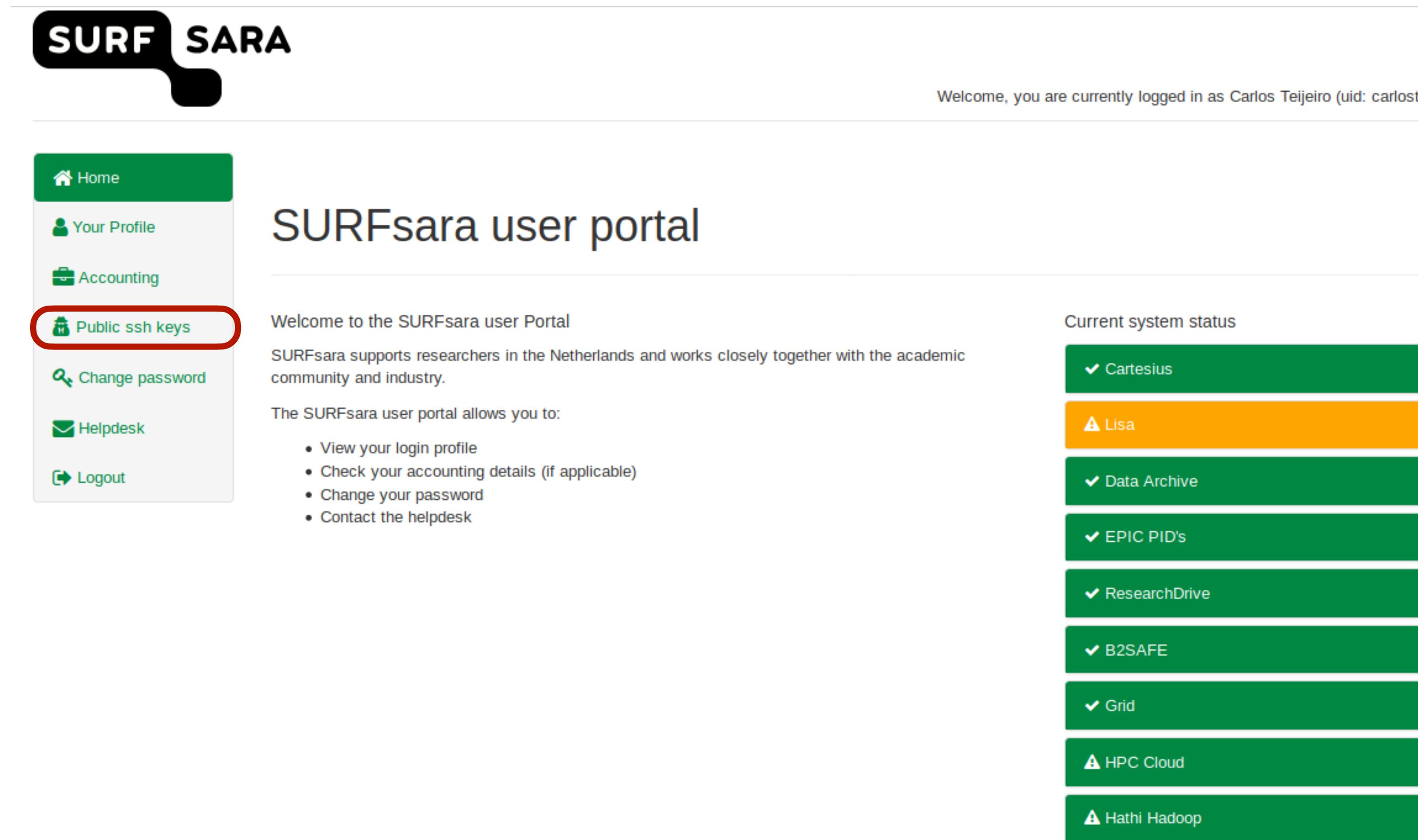
- View your login profile
- Check your accounting details (if applicable)
- Change your password
- Contact the helpdesk

Current system status

✓ Cartesius
⚠ Lisa
✓ Data Archive
✓ EPIC PID's
✓ ResearchDrive
✓ B2SAFE
✓ Grid
⚠ HPC Cloud
⚠ Hathi Hadoop

Lisa

Connecting to



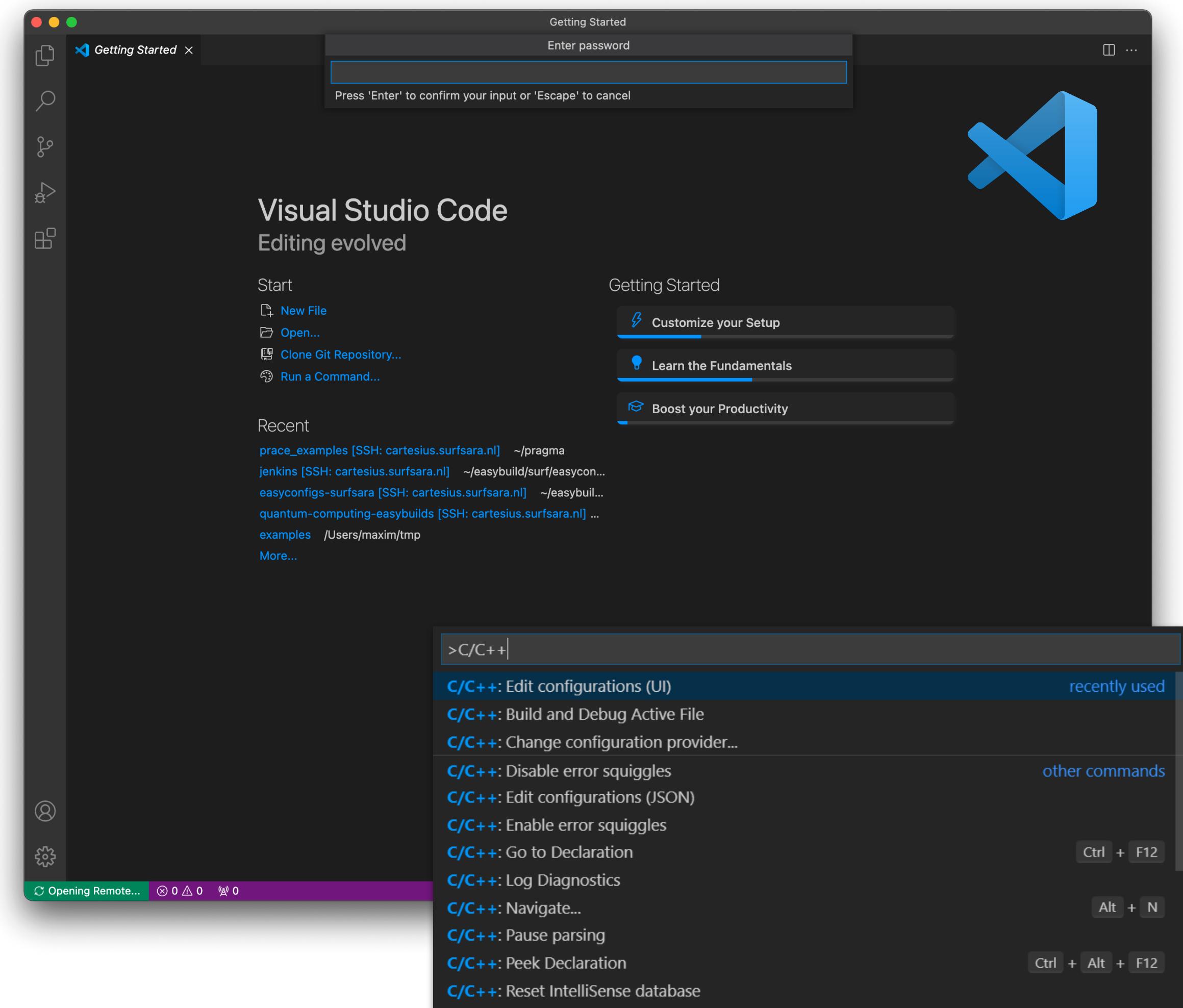
The screenshot shows the SURFsara user portal interface. At the top, there is a logo for "SURF SARA". Below it, a navigation bar includes links for "Home", "Your Profile", "Accounting", "Public ssh keys" (which is highlighted with a red oval), "Change password", "Helpdesk", and "Logout". The main content area has a heading "SURFsara user portal" and a welcome message: "Welcome to the SURFsara user Portal. SURFsara supports researchers in the Netherlands and works closely together with the academic community and industry." It also lists what the portal allows users to do: "View your login profile", "Check your accounting details (if applicable)", "Change your password", and "Contact the helpdesk". To the right, a "Current system status" section displays a list of systems with their status indicators:

System	Status
Cartesius	✓
Lisa	⚠
Data Archive	✓
EPIC PID's	✓
ResearchDrive	✓
B2SAFE	✓
Grid	✓
HPC Cloud	⚠
Hathi Hadoop	⚠

Lisa

VS Code (recommendation)

- Download VS Code from
 - <https://code.visualstudio.com>
- Install plugins
 - Locally: **Remote - SSH**
 - Remotely: **C/C++ Extension Pack**
- Press **F1** and run the **Remote-SSH: Open SSH Host...** command
- Or use any other IDE (or text editor) with support for the remote development (CLion, Eclipse, Atom, ...)



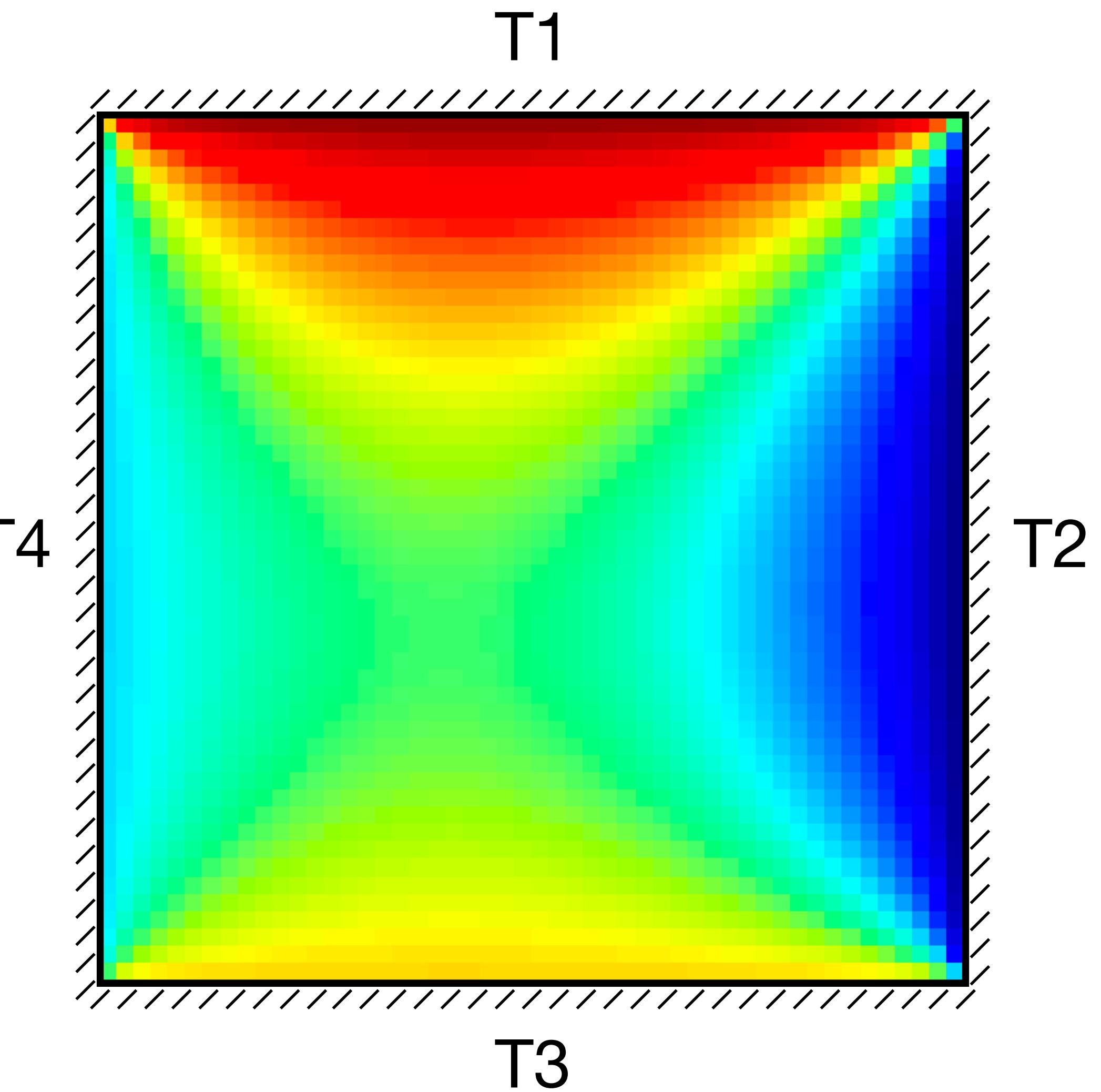
Jacobi Solver

Heat transfer

The problem

- Consider a basic steady state conduction in a closed box
- No sources are present
- All walls have constant temperature
- The governing equation:

$$\Delta T = 0 \quad \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$



Heat transfer

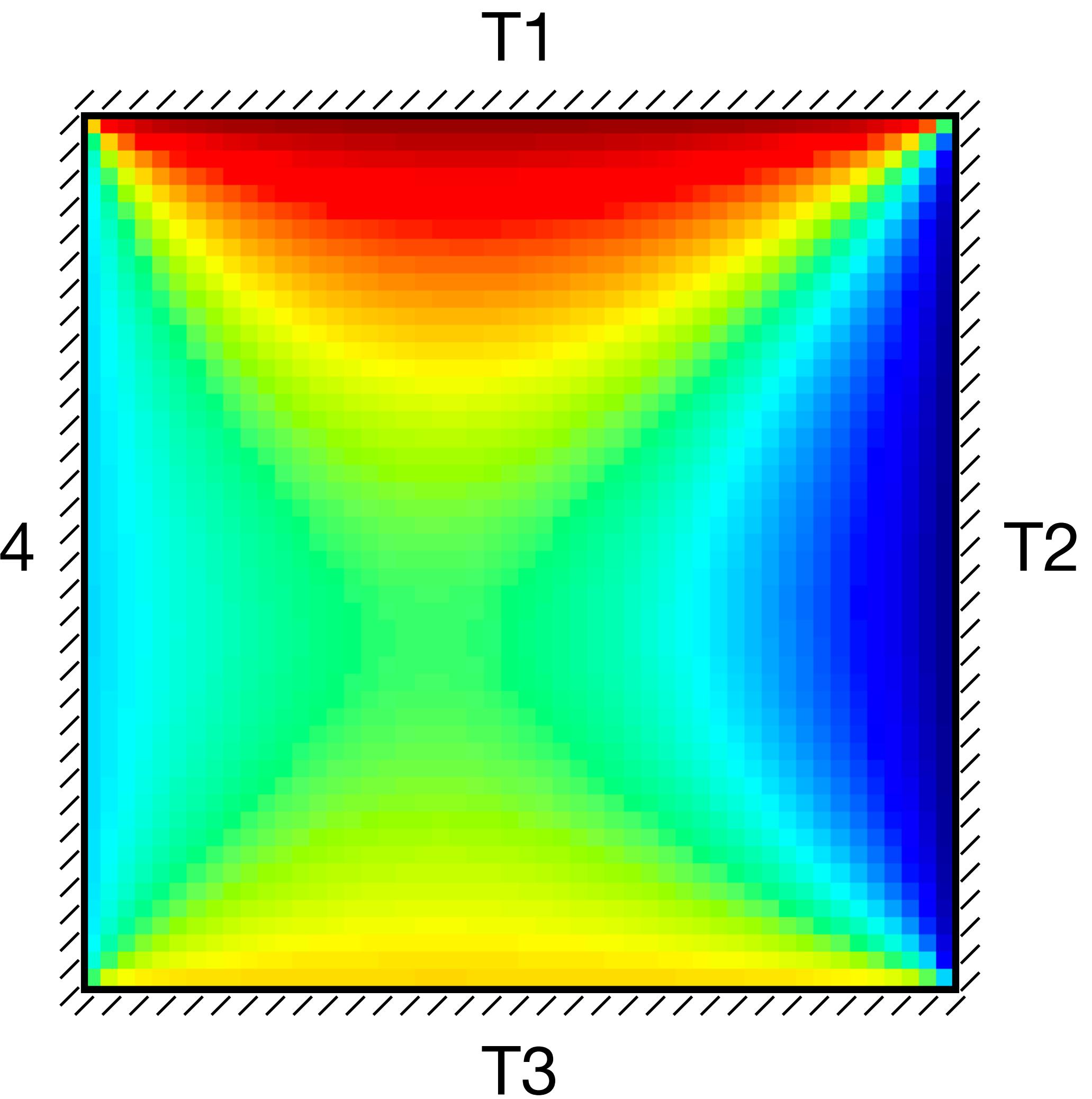
The problem

- The governing equation:

$$\Delta T = 0 \quad \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

- All derivatives are approximated with the 2nd-order central difference scheme:

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\delta x^2}$$



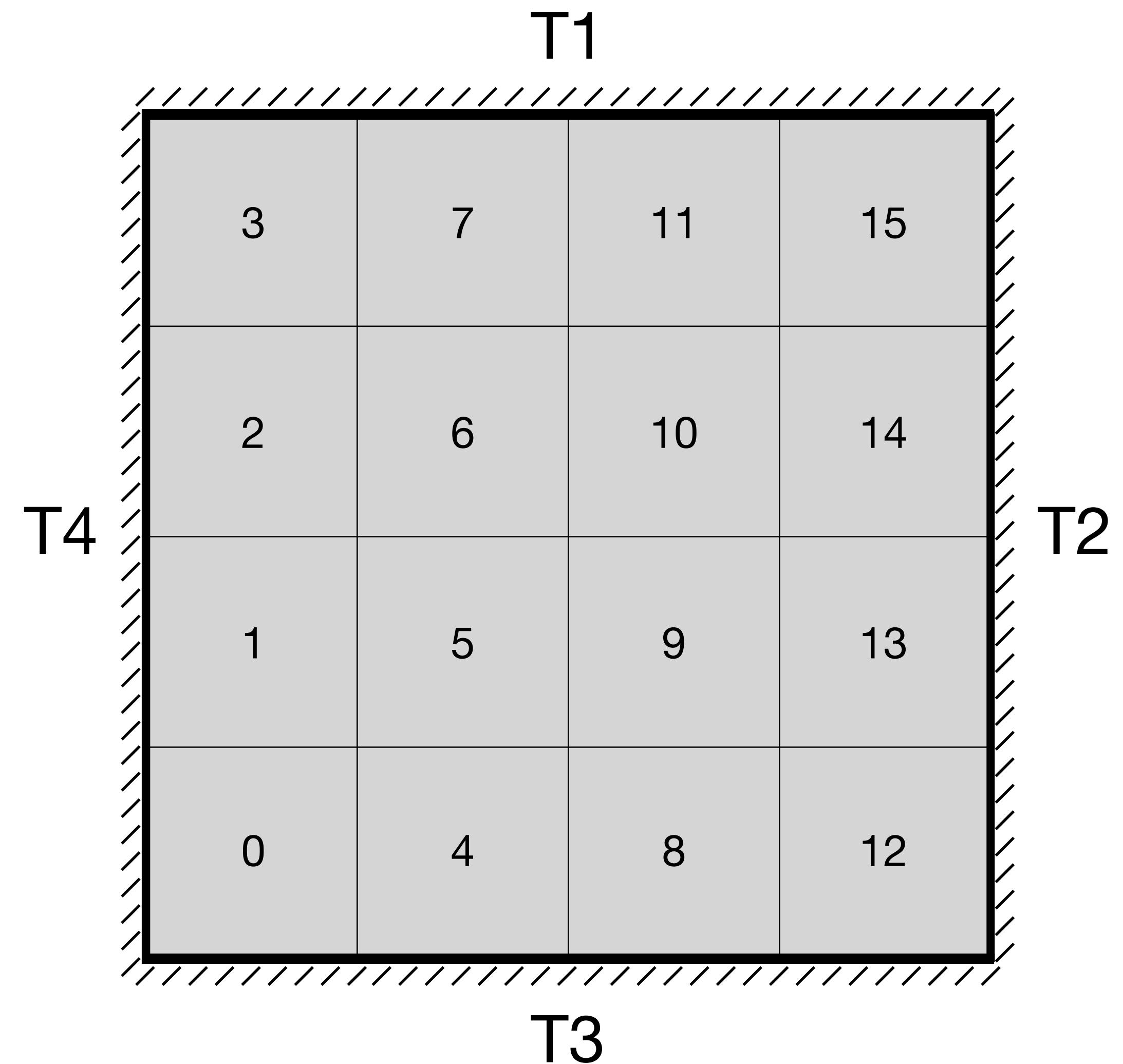
Heat transfer

The problem

- All derivatives are approximated with the 2nd-order central difference scheme:

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\delta x^2}$$

$$\frac{\partial^2 T}{\partial y^2} \approx \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\delta y^2}$$



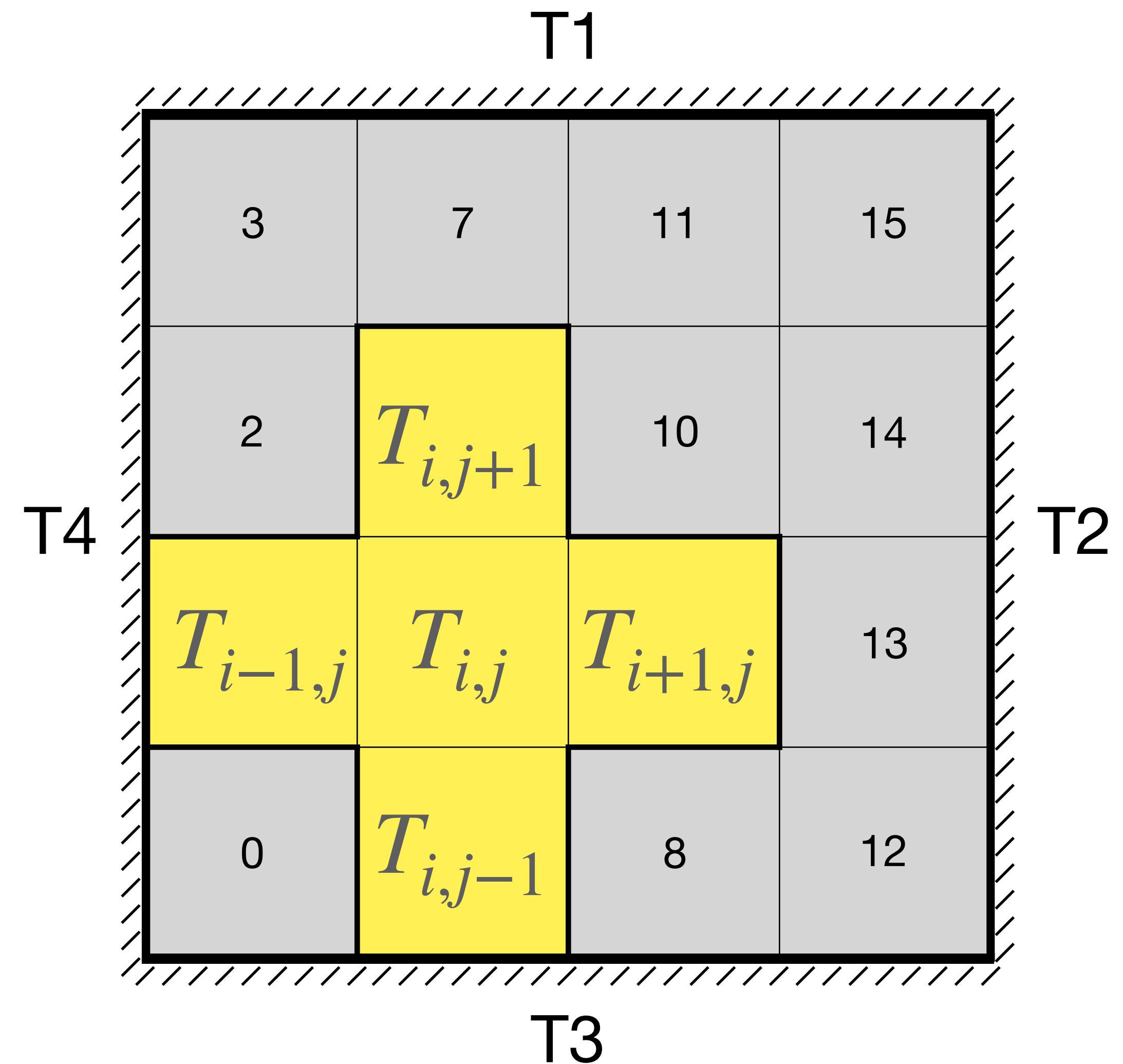
Heat transfer

The problem

- All derivatives are approximated with the 2nd-order central difference scheme:

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\delta x^2}$$

$$\frac{\partial^2 T}{\partial y^2} \approx \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\delta y^2}$$



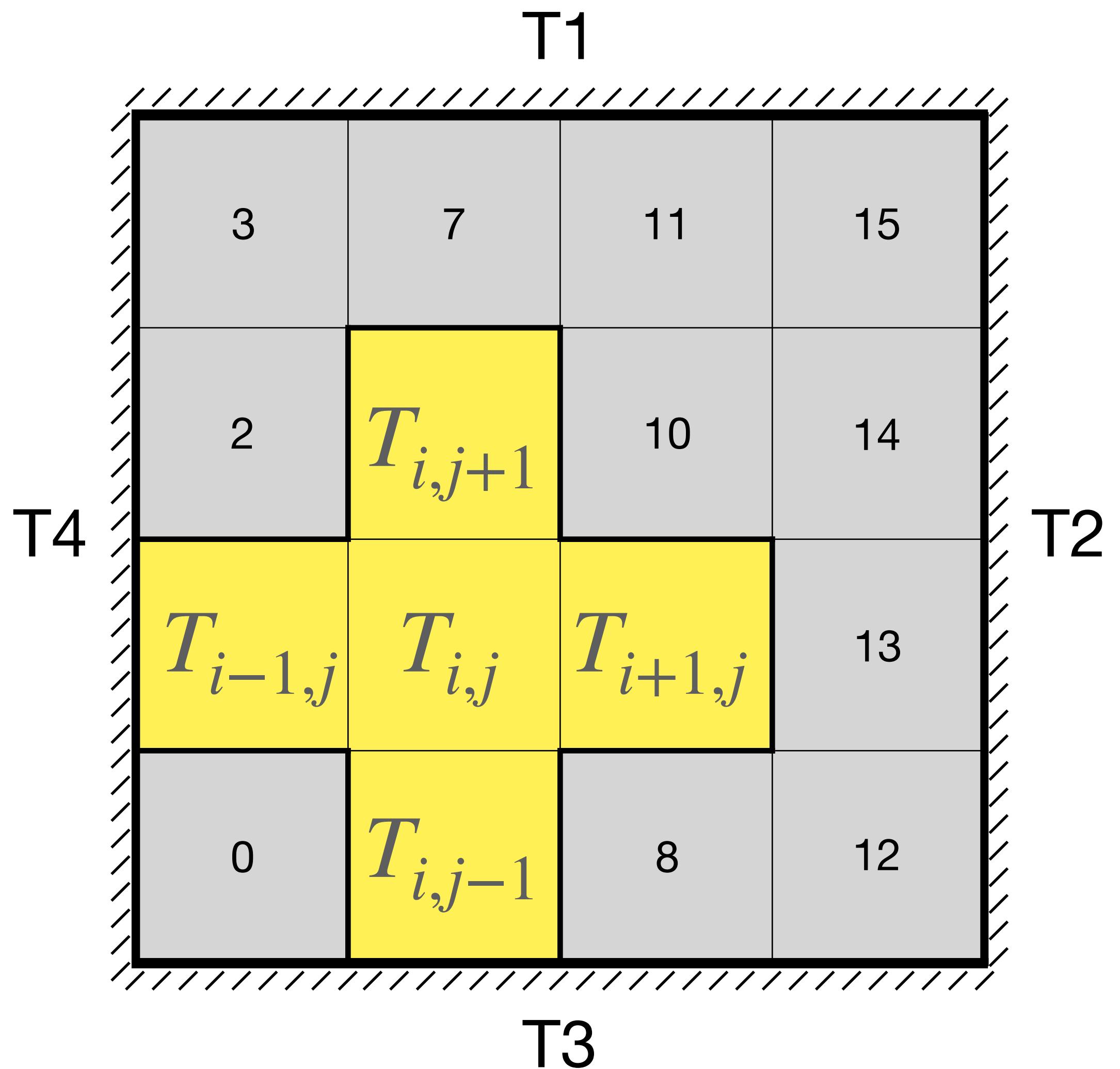
Heat transfer

The problem

- We need to solve the following system of linear equations:

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\delta x^2}$$

$$+ \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\delta y^2} = 0$$



Heat transfer

The problem

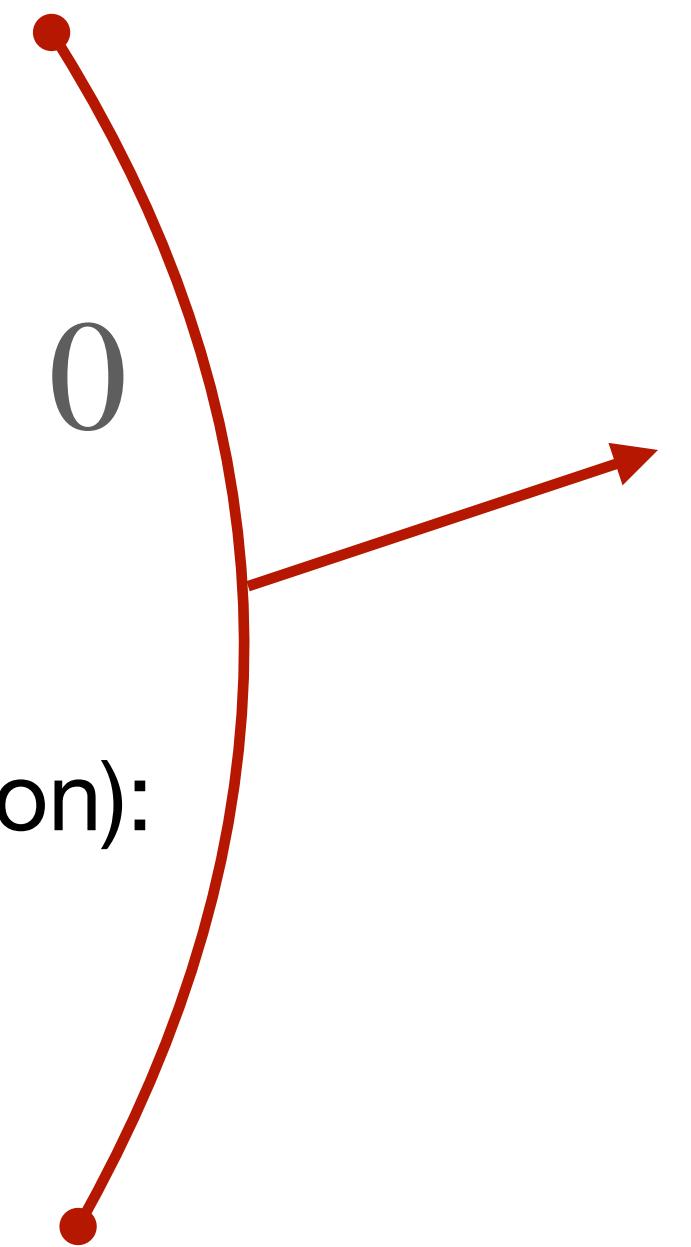
- Assume unit grid cells:

$$4T_{i,j} - T_{i+1,j} - T_{i-1,j}$$

$$-T_{i,j+1} - T_{i,j-1} = 0$$

- Boundary conditions (linear interpolation):

$$T_{i-1,j} = 2T_w - T_{i,j}$$



Matrix representation

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	6	-1			-1											
1	-1	5	-1			-1										T1
2																
3																
4	-1															
5																
6																
7																
8																
9																
10																
11																
12																
13																
14																
15																

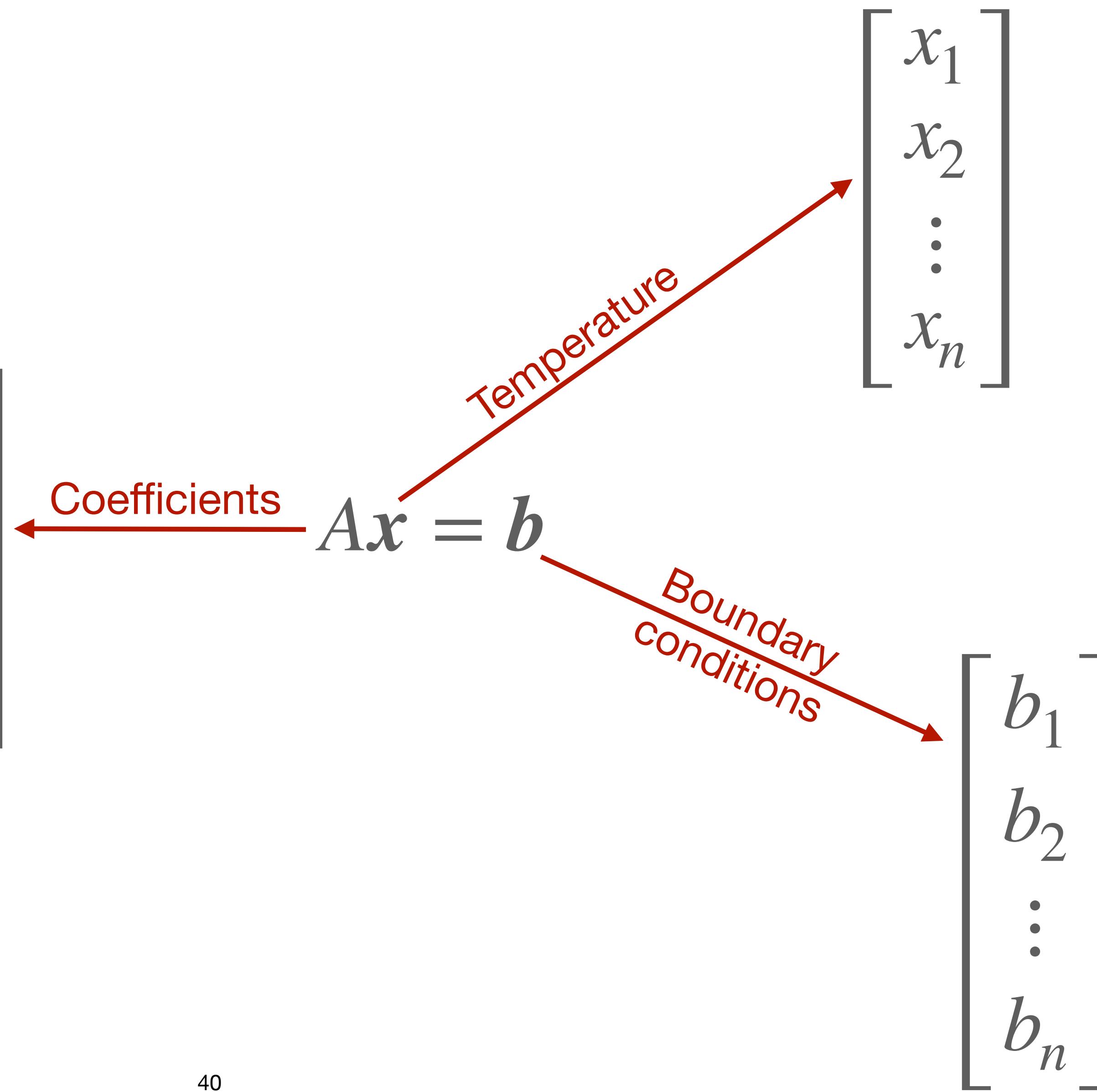
The matrix representation shows a 16x16 grid of values. The central 3x3 block is highlighted in yellow and labeled $T_{i-1,j}, T_{i,j}, T_{i+1,j}$. The row and column indices are labeled on the left and top respectively, with values from 0 to 15. Boundary conditions are indicated by arrows pointing to the first and last columns. The matrix is symmetric and tridiagonal.

T3

Heat transfer

Weighted Jacobi method

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$



Heat transfer

Weighted Jacobi method

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

The matrix A is shown as a 4x4 grid of elements a_{ij} . The main diagonal elements a_{ii} are highlighted in blue and labeled D . The sub-diagonal elements a_{ij} for $j < i$ are highlighted in red and labeled L . The super-diagonal elements a_{ij} for $j > i$ are highlighted in green and labeled U .

$$A = D + L + U$$

$$\begin{aligned} \mathbf{x}^{(k+1)} = & \omega D^{-1}(\mathbf{b} - (L + U)\mathbf{x}^{(k)}) \\ & + (1 - \omega)\mathbf{x}^{(k)} \end{aligned}$$

$$\begin{aligned} x_i^{(k+1)} = & \frac{1}{a_{ii}}(b_i - \sum_{j \neq i} a_{ij}x_j^{(k)}) \\ & + (1 - \omega)x_i^{(k)} \end{aligned}$$

Heat transfer

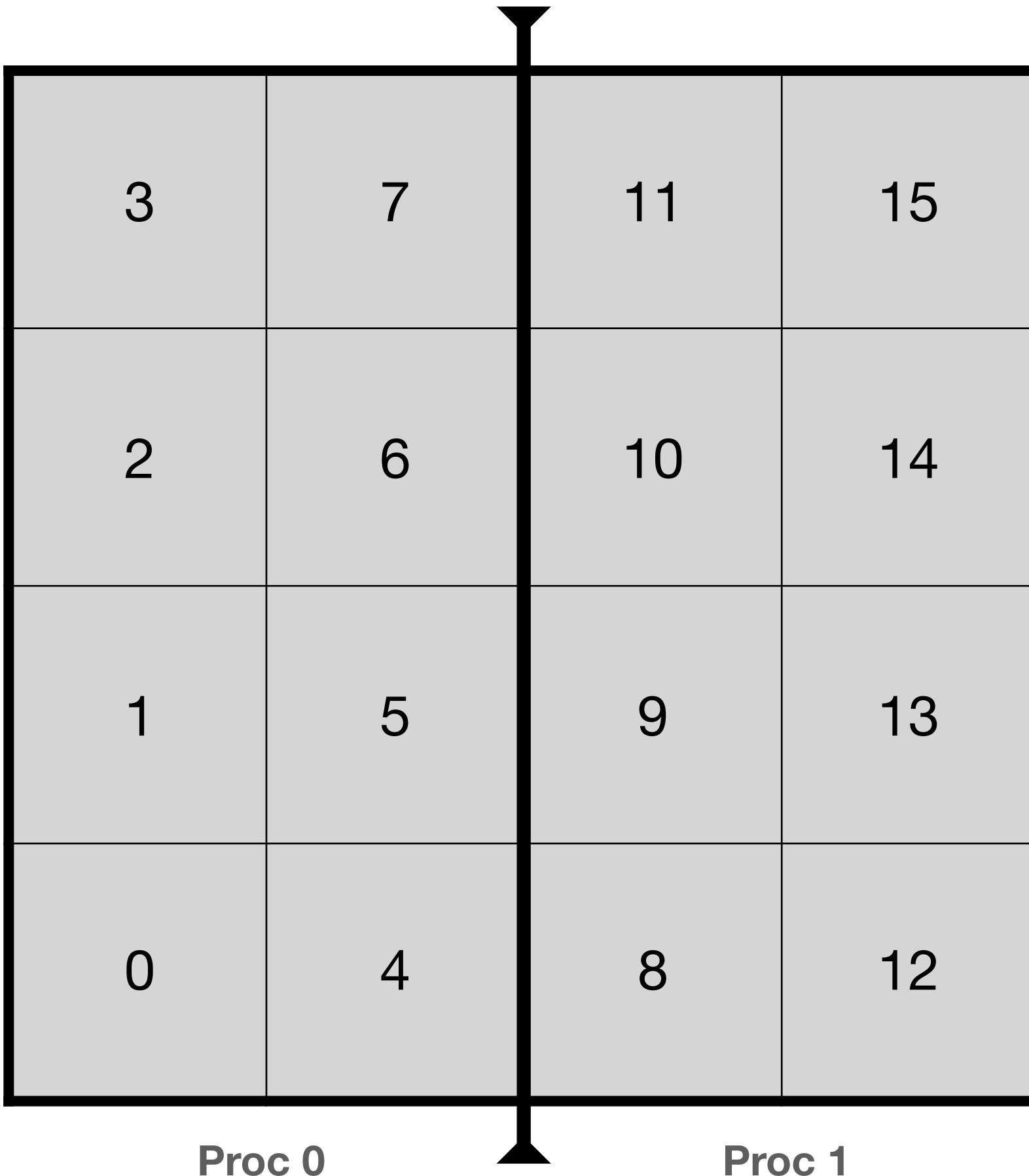
Why weighted Jacobi method?

- Simple implementation
- Has simple but frequently used mathematical operations:
 - **L2-norm**
 - **MVP**
 - **Vector update (copy)**
- Simple parallelisation
- Straightforward hybrid parallelisation
- Often used as an addition in more sophisticated linear solvers (e.g. as a smoother in AMG)

Heat transfer

Data representation

Grid representation



Matrix representation

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	6	-1			-1											
1	-1	5	-1			-1										
2		-1	5	-1			-1									
3			-1	6				-1								
4	-1				5	-1			-1							
5		-1			-1	4	-1			-1						
6			-1			-1	4	-1			-1					
7				-1			-1	5			-1					
8					-1			5	-1			-1				
9						-1			-1	4	-1		-1			
10							-1			-1	4	-1		-1		
11								-1			-1	5			-1	
12									-1			6	-1			
13										-1			-1	5	-1	
14											-1			-1	5	-1
15												-1			-1	6

Heat transfer

Data representation

Grid representation



Matrix representation

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	6	-1			-1											
1	-1	5	-1			-1										
2		-1	5	-1			-1									
3			-1	6				-1								
4	-1			5	-1				-1							
5		-1		-1	4	-1				-1						
6			-1		-1	4	-1				-1					
7					-1		-1	5				-1				
8						-1			5	-1		-1				
9						-1			-1	4	-1		-1			
10							-1		-1	4	-1		-1			
11							-1		-1	-1	5			-1		
12								-1			6	-1				
13									-1		-1	5	-1			
14									-1		-1	-1	5	-1		
15										-1		-1	-1	5	-1	

MVP

Data representation

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Proc 0	0	6	-1			-1										
1	-1	5	-1			-1										
2		-1	5	-1			-1									
3			-1	6				-1								
4	-1				5	-1			-1							
5		-1			-1	4	-1			-1						
6			-1			-1	4	-1			-1					
7				-1			-1	5			-1					
8					-1				5	-1			-1			
9						-1			-1	4	-1			-1		
10							-1			-1	4	-1			-1	
11								-1		-1	5				-1	
12								-1				6	-1			
13									-1		-1	5	-1			
14										-1		-1	5	-1		
15											-1		-1	6		

$$x = \begin{matrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{15} \end{matrix} = \begin{matrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \\ y_{12} \\ y_{13} \\ y_{14} \\ y_{15} \end{matrix}$$

$$y_i = \sum_{j=0}^N a_{ij} x_j$$

$$\begin{aligned} y_5 &= a_{51}x_1 + a_{54}x_4 \\ &\quad + a_{55}x_5 + a_{56}x_6 \\ &\quad + a_{59}x_9 \end{aligned}$$

MVP

Data representation

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Proc 0	6	-1			-1											
1	-1	5	-1			-1										
2		-1	5	-1			-1									
3			-1	6				-1								
4	-1				5	-1			-1							
5		-1			-1	4	-1			-1						
6			-1			-1	4	-1			-1					
7				-1			-1	5				-1				
8					-1				5	-1			-1			
9						-1			-1	4	-1			-1		
10							-1			-1	4	-1			-1	
11								-1			-1	5			-1	
12									-1			6	-1			
13										-1		-1	5	-1		
14											-1		5	-1		
15												-1		6		

$$\begin{array}{c}
 \begin{matrix} & x_0 & & & & y_0 \\ & x_1 & & & & y_1 \\ & x_2 & & & & y_2 \\ & x_3 & & & & y_3 \\ & x_4 & & & & y_4 \\ & x_5 & & & & y_5 \\ & x_6 & & & & y_6 \\ & x_7 & & & & y_7 \\ & x_8 & & & & y_8 \\ & x_9 & & & & y_9 \\ & x_{10} & & & & y_{10} \\ & x_{11} & & & & y_{11} \\ & x_{12} & & & & y_{12} \\ & x_{13} & & & & y_{13} \\ & x_{14} & & & & y_{14} \\ & x_{15} & & & & y_{15} \end{matrix} \\
 = \\
 \begin{matrix} x \\ y \end{matrix}
 \end{array}$$

X

We need access to the value from the neighbouring process!

$$y_i = \sum_{j=0}^N a_{ij} x_j$$

$$y_5 = a_{51}x_1 + a_{54}x_4$$

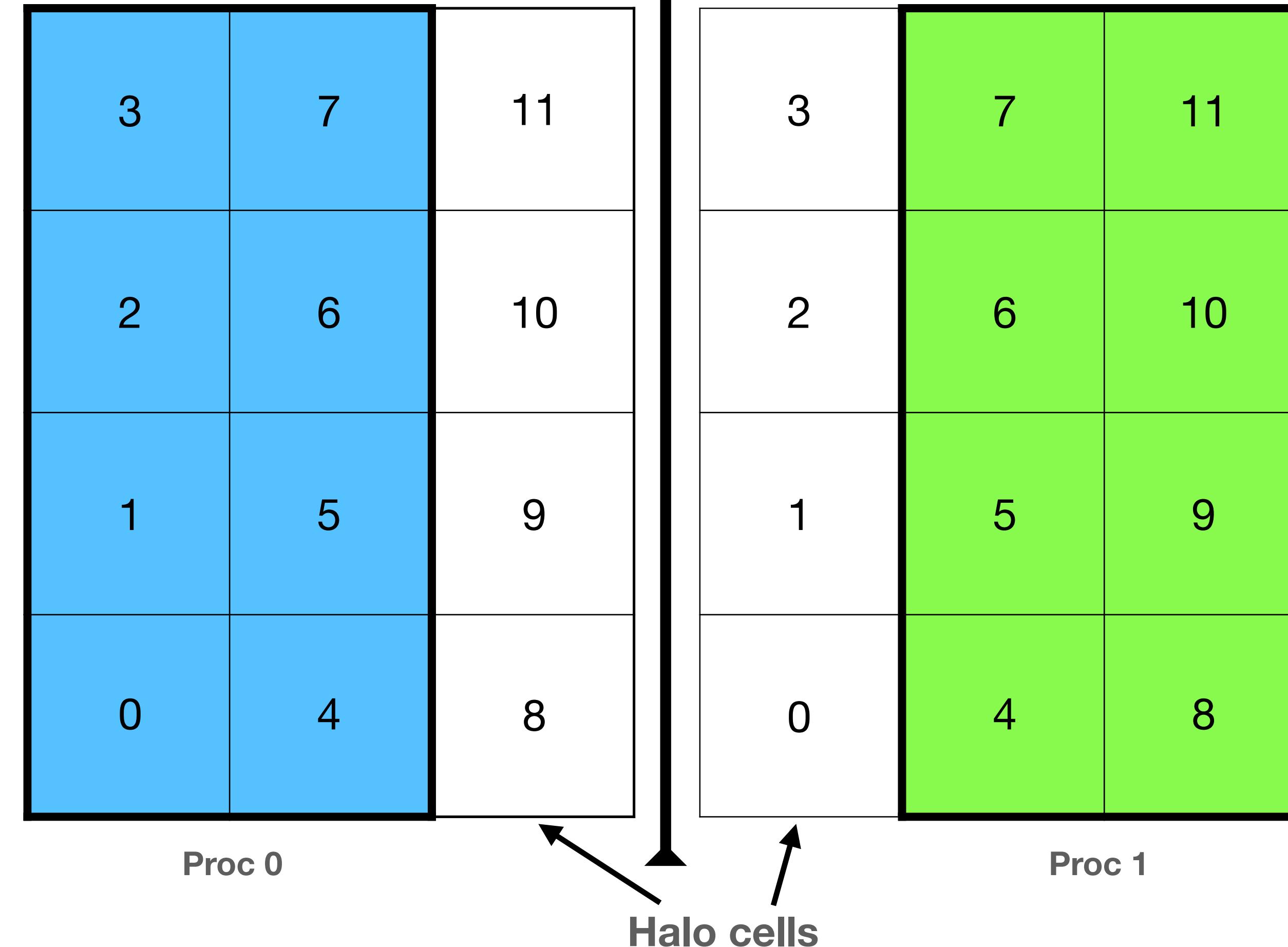
$$+ a_{55}x_5 + a_{56}x_6$$

$$+ a_{59}x_9$$

Heat transfer

Data representation

Grid representation



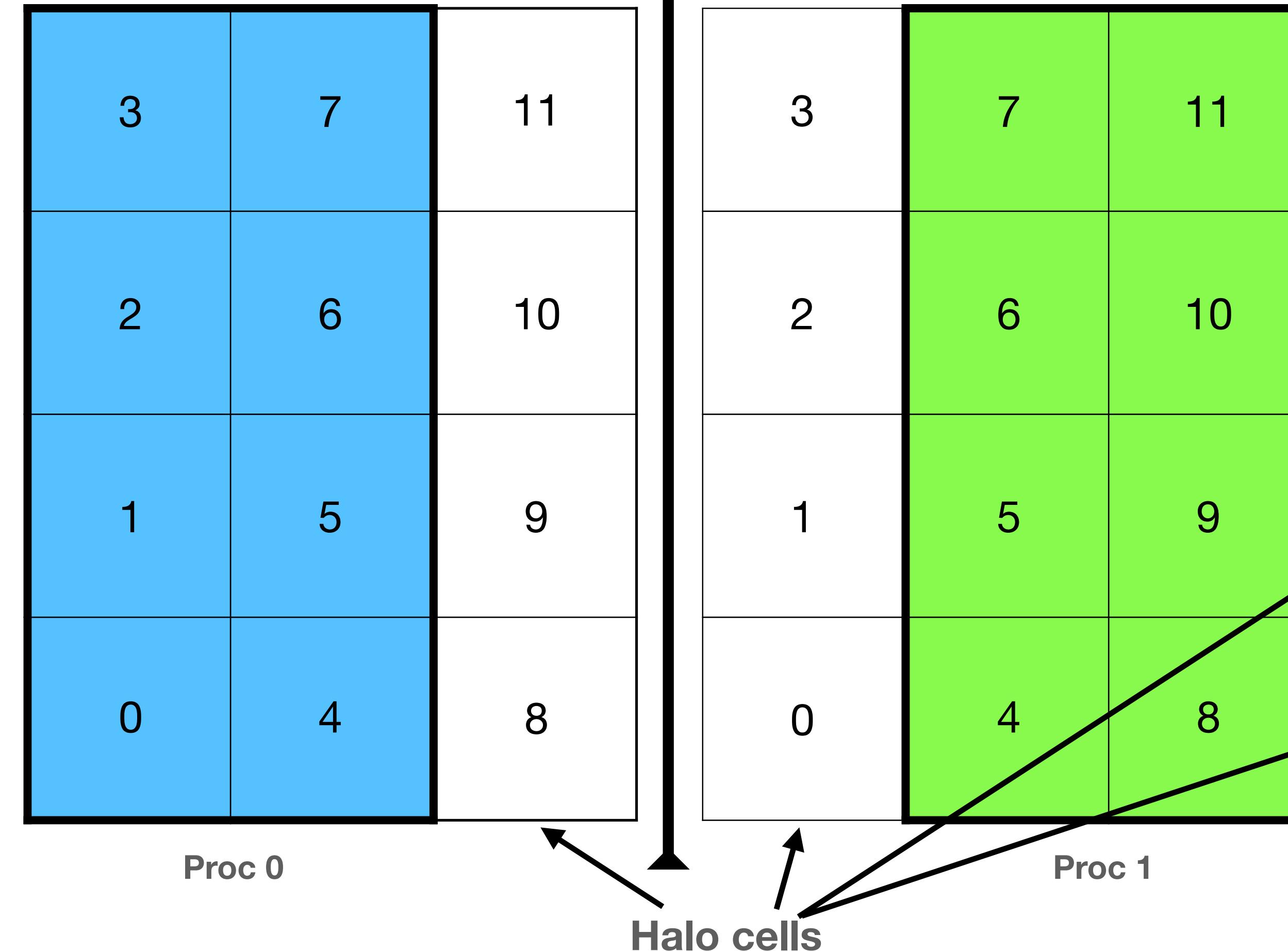
Matrix representation

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	6	-1			-1											
1	-1	5	-1			-1										
2		-1	5	-1			-1									
3			-1	6				-1								
4	-1				5	-1			-1							
5		-1			-1	4	-1			-1						
6			-1			-1	4	-1			-1					
7				-1			-1	5			-1					
8					-1			5	-1			-1				
9						-1			-1	4	-1		-1			
10							-1			-1	-1	4	-1		-1	
11								-1			-1	-1	5		-1	
12									-1			-1	6	-1		
13										-1		-1	-1	5		-1
14										-1		-1	-1	-1	5	-1
15											-1		-1	-1	-1	6

Heat transfer

Data representation

Grid representation



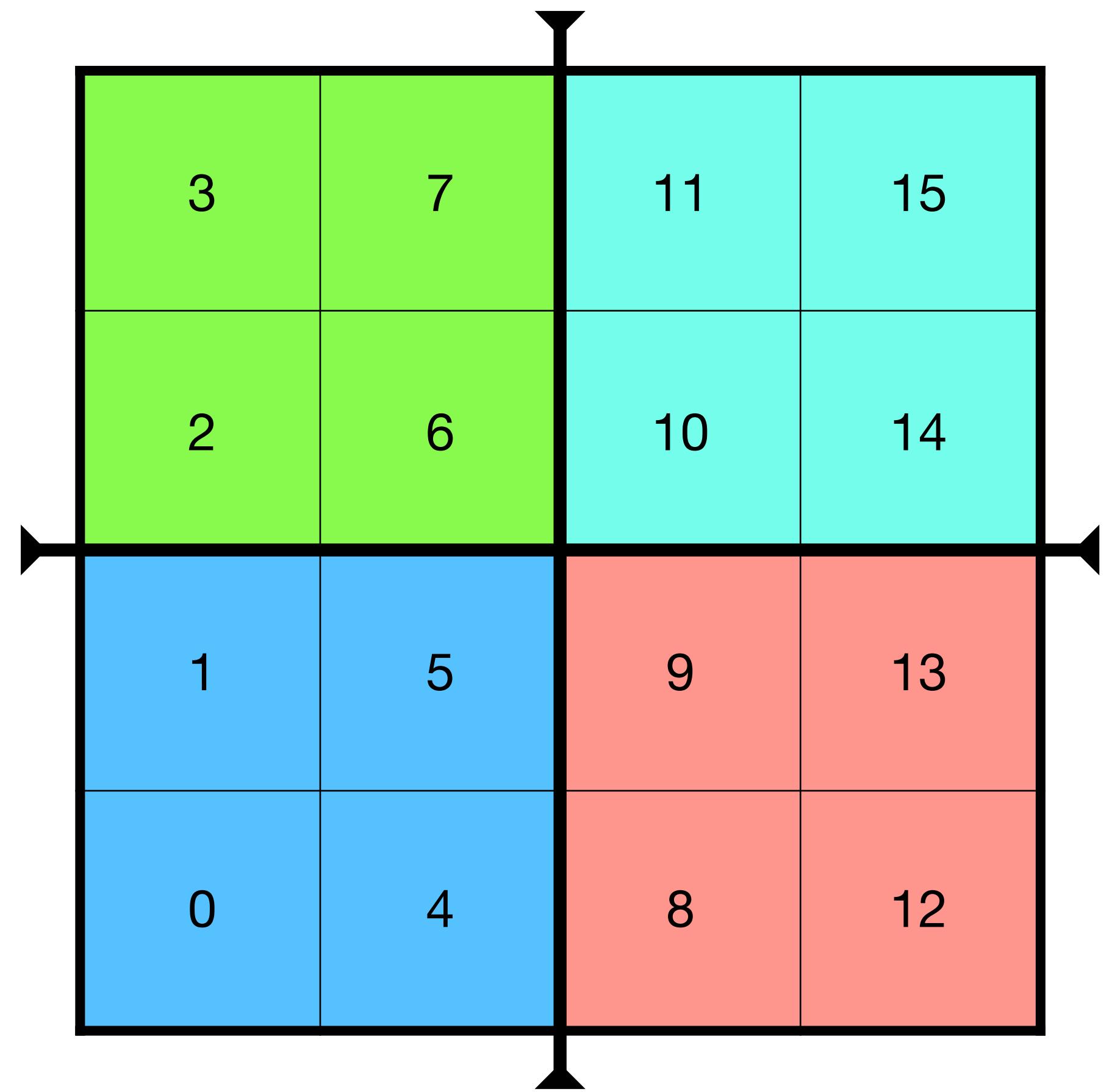
Matrix representation

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	6	-1			-1											
1	-1	5	-1			-1										
2		-1	5	-1			-1									
3			-1	6				-1								
4	-1			5	-1				-1							
5		-1		-1	4	-1				-1						
6			-1	-1	4	-1					-1					
7				-1	-1	5						-1				
8					-1			5	-1			-1				
9						-1		-1	4	-1			-1			
10							-1	-1	-1	4	-1			-1		
11								-1		-1	5				-1	
12									-1			-1	4	-1		
13										-1		-1	5			
14										-1			-1	6	-1	
15											-1		-1	5	-1	

No problems far!

Heat transfer

Data representation

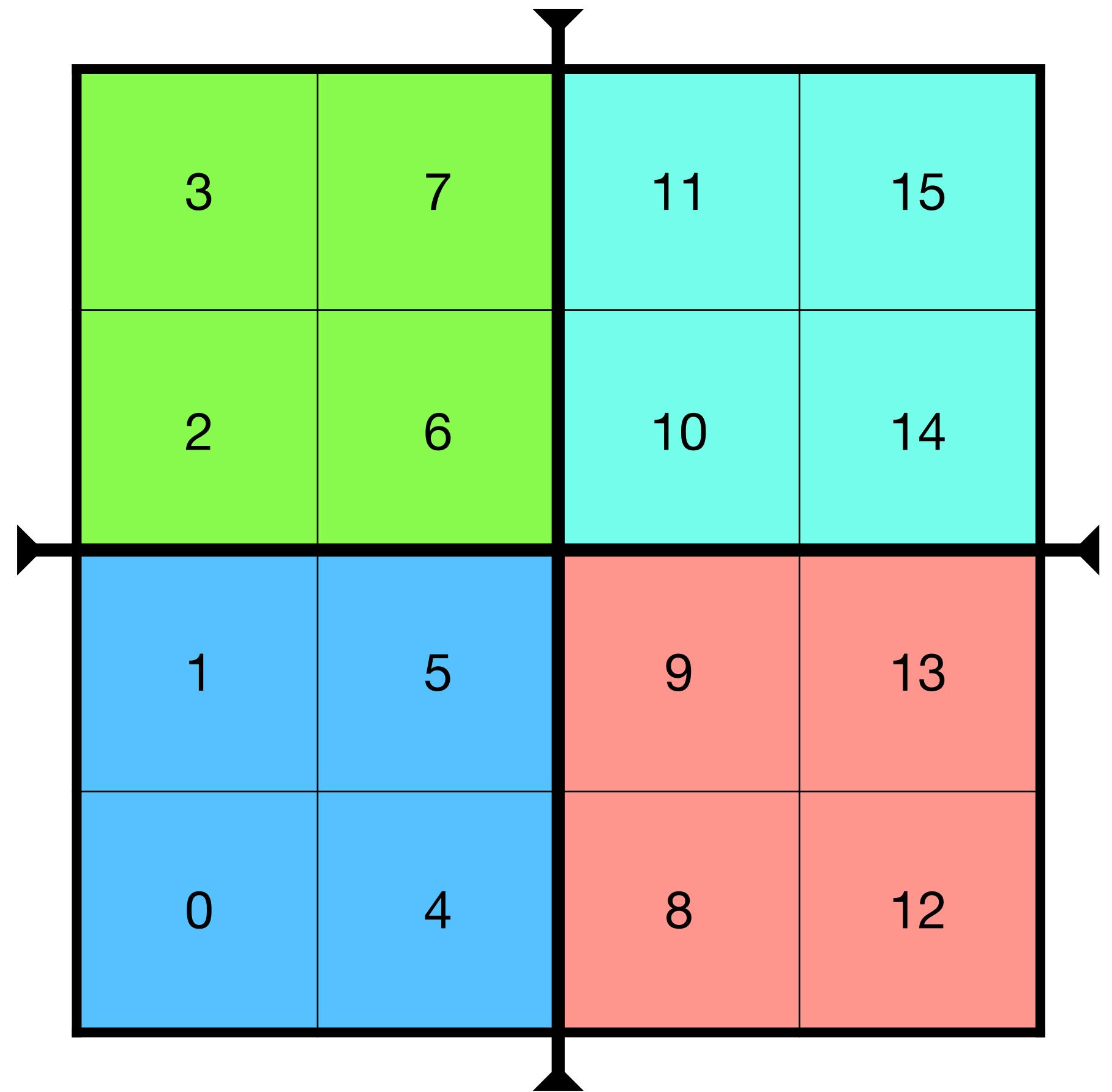


Matrix representation

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	6	-1			-1											
1	-1	5	-1			-1										
2		-1	5	-1			-1									
3			-1	6				-1								
4	-1				5	-1			-1							
5		-1			-1	4	-1			-1						
6			-1			-1	4	-1			-1					
7				-1			-1	5				-1				
8					-1				5	-1			-1			
9						-1			-1	4	-1			-1		
10							-1			-1	4	-1			-1	
11								-1		-1	-1	5			-1	
12									-1				6	-1		
13										-1			-1	5	-1	
14											-1			-1	5	-1
15												-1		-1	-1	6

Heat transfer

Data representation



Matrix representation

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	6	-1			-1											
1	-1	5	-1			-1										
2		-1	5	-1			-1									
3			-1	6				-1								
4	-1				5	-1			-1							
5		-1			-1	4	-1			-1						
6			-1			-1	4	-1			-1					
7				-1			-1	5				-1				
8					-1				5	-1			-1			
9						-1			-1	4	-1			-1		
10							-1			-1	4	-1			-1	
11								-1			-1	5			-1	
12									-1			6	-1			
13										-1			-1	5	-1	
14										-1			-1	5	-1	
15											-1			-1	6	

Heat transfer

Data representation

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	6	-1			-1											
1	-1	5	-1			-1										
2		-1	5	-1			-1									
3			-1	6				-1								
4	-1			5	-1				-1							
5		-1			-1	4	-1			-1						
6			-1			-1	4	-1			-1					
7				-1			-1	5				-1				
8					-1			5	-1			-1				
9						-1	4	-1				-1				
10							-1		-1	4	-1			-1		
11								-1			-1	5				-1
12									-1			6	-1			
13										-1			-1	5	-1	
14										-1				-1	5	-1
15											-1				-1	6

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	6	-1			-1											
1	-1	5	-1			-1										
2		-1	5	-1			-1									
3			-1	6				-1								
4	-1			5	-1				-1							
5		-1			-1	4	-1			-1						
6			-1			-1	4	-1			-1					
7				-1			-1	5				-1				
8					-1			5	-1			-1				
9						-1	4	-1				-1	4	-1		
10							-1		-1	4	-1				-1	
11								-1			-1	5				-1
12									-1			6	-1			
13										-1			-1	5	-1	
14										-1				-1	5	-1
15											-1				-1	6



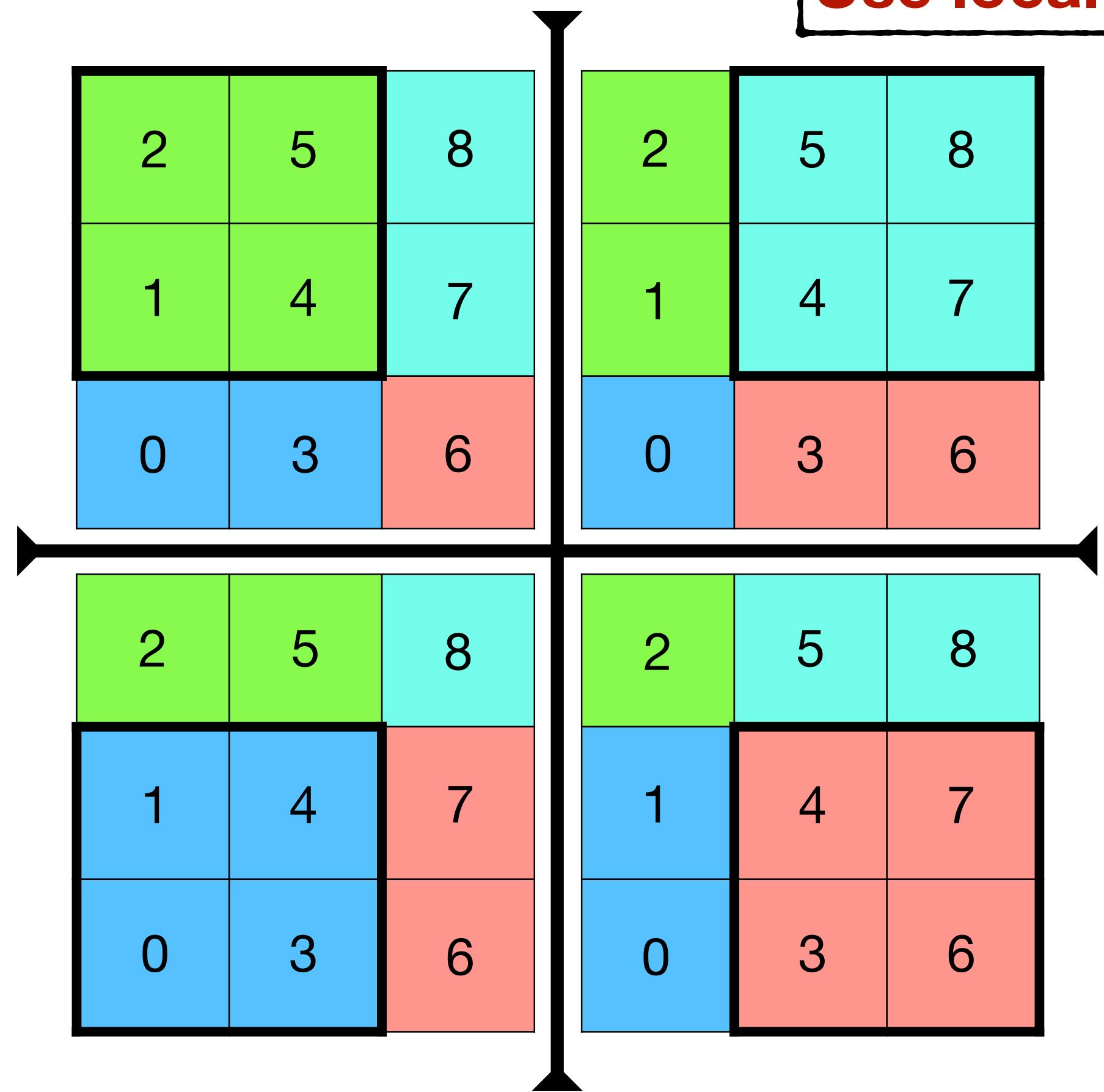
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	0	-1	5	-1			-1									
3			-1	6				-1								
6		-1			-1	4	-1			-1						
7			-1			-1	5			-1						
8				-1			5	-1			-1					
9					-1			-1	4	-1		-1	4	-1		
12									-1			-1		6	-1	
13										-1			-1	-1	5	-1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8					-1					5	-1					
9						-1				-1	4	-1				
10											-1	4	-1			
11											-1	5				
12												-1		6	-1	
13													-1	5	-1	
14													-1	5	-1	
15													-1	6		

Heat transfer

Data representation

**Optimised layout!
Use local indices!**



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	6	-1		-1												
1	-1	5	-1		-1											
2	-1			5	-1		-1									
3		-1		-1	4	-1		-1								

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	-1	5	-1		-1											
1		-1	6			-1										
2		-1		-1	4	-1		-1								
3			-1		-1	5			-1							

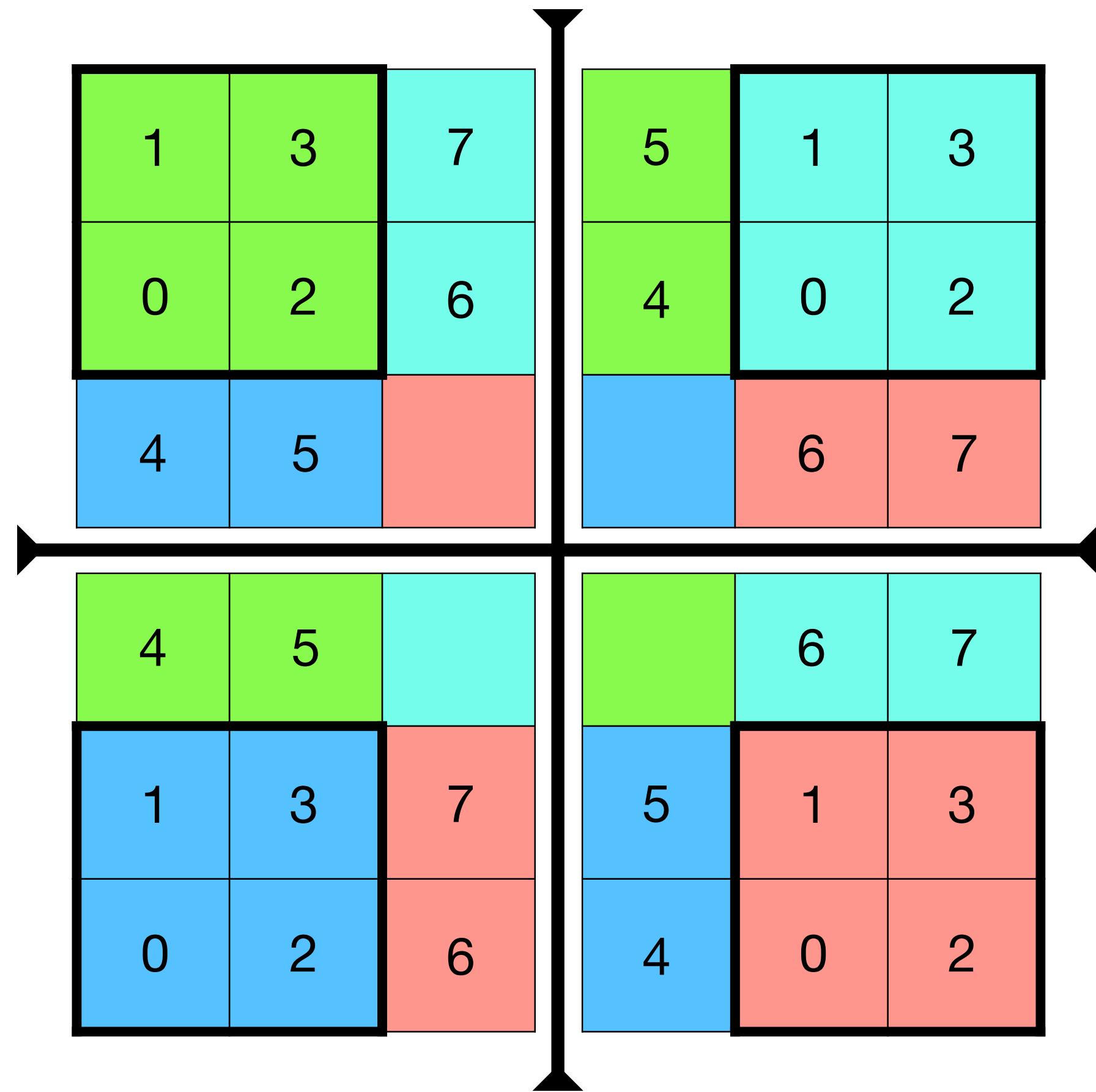
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	-1			5	-1		-1									
1		-1		-1	4	-1		-1								
2				-1			6	-1								
3					-1		-1	5	-1							

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0		-1		-1	4	-1		-1								
1			-1		-1		-1	5			-1					
2					-1		-1	-1	5	-1						
3						-1		-1	-1	5	-1					

Heat transfer

Data representation

Re-enumereation!



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	6	-1	-1													
1	-1	5		-1	-1											
2	-1		5	-1												
3		-1	-1	4		-1										

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	5	-1	-1		-1											
1	-1	6		-1												
2	-1		4	-1		-1	-1									
3		-1	-1	5												

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	5	-1	-1		-1											
1	-1	4		-1		-1	-1									
2	-1		6	-1												
3		-1	-1	5												

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	-1	-1		-1											
1	-1	5		-1		-1										
2	-1		5	-1												
3		-1	-1	6												

Heat transfer

Data representation

Optimised storage!

Memory reduction (~x2):

- Regular storage: $N_{glob} \times N_{loc}$
- Modified storage: $N_{loc} \times (N_{loc} + N_{halo})$

Easier access to halo values

Compact storage of the “local” values

In many numerical codes the halo values are stored in a separate array

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	6	-1	-1													
1	-1	5		-1	-1											
2	-1		5	-1					-1							
3		-1	-1	4		-1			-1							

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	5	-1	-1		-1											
1	-1	6		-1												
2	-1		4	-1		-1	-1									
3		-1	-1	5					-1							

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	5	-1	-1		-1											
1	-1	4		-1		-1	-1									
2	-1		6	-1												
3		-1	-1	5					-1							

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	-1	-1		-1											
1	-1	5		-1		-1										
2	-1		5	-1												
3		-1	-1	6												

Heat transfer

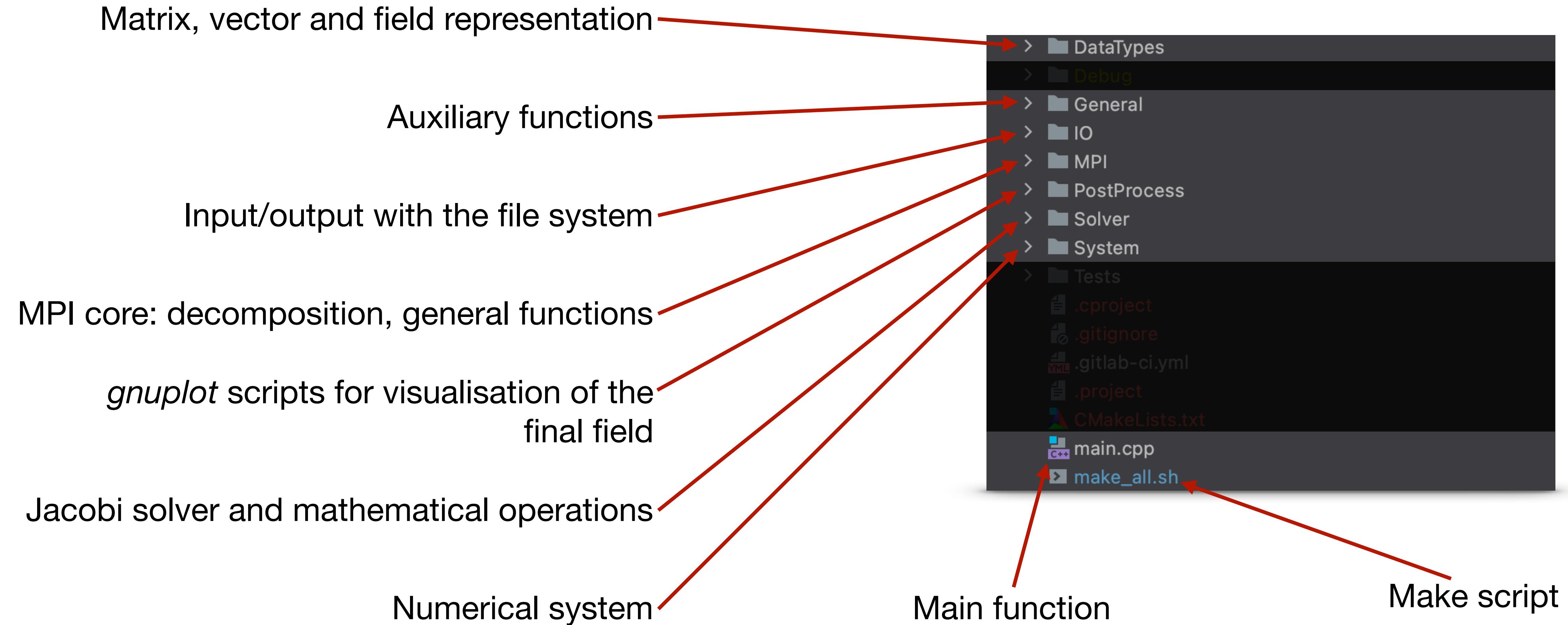
Repository

- SSH to Lisa
- Clone the repository from: **https://github.com/sara-nl/prace_jacobi.git**
- Compile with `./make_all.sh <type>` (run `./make_all.sh` to see all available types)

```
$ git clone https://github.com/sara-nl/prace_jacobi.git
$ cd prace_jacobi
$ ./make_all.sh mpi
```

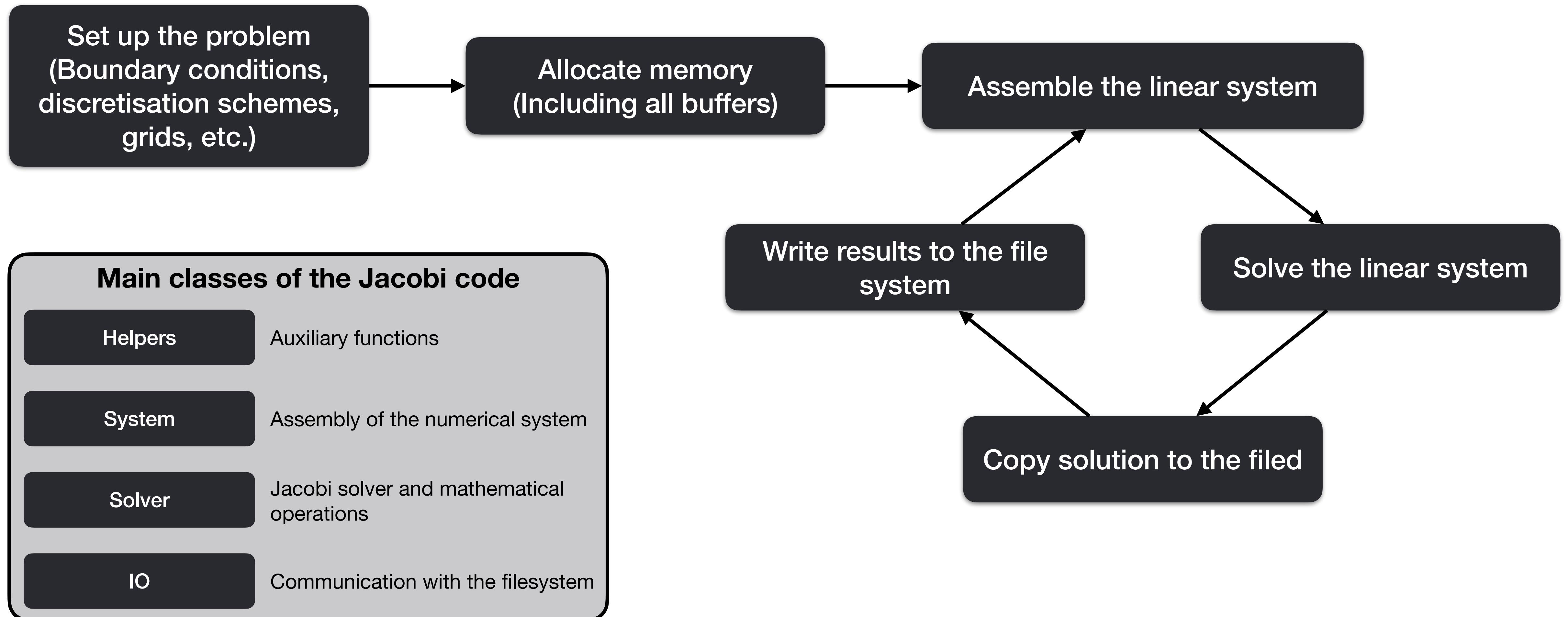
Heat transfer

Code structure



Heat transfer

Generic logic



Hands-on

Hands-on

Filling the empty space

- Implement in **Solver/solver.cpp** (look for the **NOT_IMPLEMENTED** macro):
 - Vector copy **Solver::copyVector()**
 - Residual calculation
Solver::calculateResidual()
 - L2-norm calculation **Solver::calculateNorm()**
- Test your implementation by compiling with `omp` flag (will still be serial)
- Run the executable without any arguments
- Visualise results with *gnuplot*

```
void Solver::copyVector(Vector &vec_in, Vector &vec_out) {  
    /*  
     * for every n-th elements in `vec_in`  
     *      assign element of vec_in(n) to vec_out(n)  
     */  
    NOT_IMPLEMENTED  
}  
  
void Solver::calculateResidual(Matrix &A, Vector &x,  
                               Vector &b, Vector &res) {  
    /*  
     * assign `b` to `res`  
     *  
     * for vector `x` and matrix `A`, the residual `res`  
     * is calculated as:  
     *      res(i) = sum( A(i, j) * x(j) )  
     */  
    NOT_IMPLEMENTED  
}  
  
double Solver::calculateNorm(Vector &vec) {  
    /*  
     * for vector `vec` with n elements  
     *      L2-norm = sqrt( sum( vec(n) * vec(n) ) )  
     *                  |  
     *                  | local for every MPI process  
     *                  |  
     *                  |  
     * one value for all MPI processes  
     */  
    NOT_IMPLEMENTED  
}  
    return 0.0;  
}
```

Hands-on

Filling the empty space

- Class **Vector** is inherited from class **Matrix**
- Most important methods:

```
/*
 * Get number of local rows
 * (including halo cells)
 */
Matrix:: numRows();

/*
 * Get number of local columns
 * (including halo cells)
 */
Matrix:: numCols();

/*
 * Get number of local elements
 * (excluding halo cells)
 */
Matrix:: getLocElts();
```

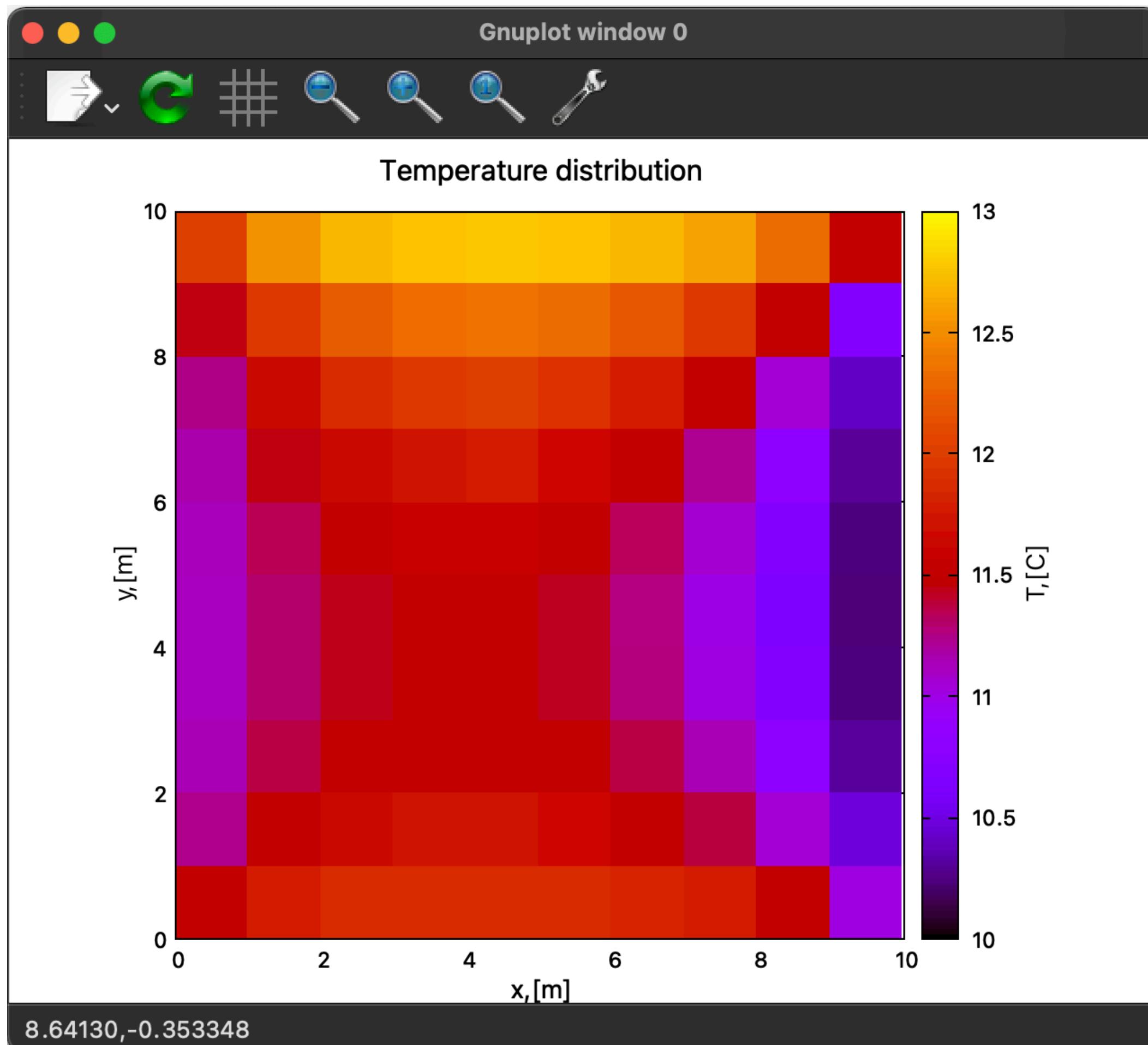
```
void Solver:: copyVector(Vector &vec_in, Vector &vec_out) {
    /*
     * for every n-th elements in `vec_in`
     * assign element of vec_in(n) to vec_out(n)
     */
    NOT_IMPLEMENTED
}

void Solver:: calculateResidual(Matrix &A, Vector &x,
                                 Vector &b, Vector &res) {
    /*
     * assign `b` to `res`
     *
     * for vector `x` and matrix `A`, the residual `res`
     * is calculated as:
     *     res(i) = sum( A(i, j) * x(j) )
     */
    NOT_IMPLEMENTED
}

double Solver:: calculateNorm(Vector &vec) {
    /*
     * for vector `vec` with n elements
     * L2-norm = sqrt( sum( vec(n) * vec(n) ) )
     *           | \_____| / |
     *           | local for every MPI process | /
     *           \_____| / one value for all MPI processes
     */
    NOT_IMPLEMENTED
    return 0.0;
}
```

Hands-on

Build and execution



```
$ module load 2020 foss/2020a
$ ./make_all.sh omp
$
$ cat first_run.sh
#!/bin/bash
#SBATCH --reservation=mpi_advanced
#SBATCH -n 1
#SBATCH --job-name=first_run
#SBATCH --output=first_run.out
#SBATCH --error=first_run.err
#SBATCH --time=0:05:00
```

Job script

```
srun ./a.out -s 10 10 -d 1 1
$
$ sbatch first_run.sh
```

```
$ cat first_run.out
0      0.588603
1      0.392595
...
352    1.02557e-06
353    9.92268e-07
Writing results to file: output.dat
Elapsed time (Jacobi): 0.005982s.
Elapsed time (I0): 0.012528s.
$
$ # if used "ssh -X" or "ssh -Y"
$ cp PostProcess/plot_omp.plt .
$ gnuplot
gnuplot> load "plot_omp.plt"
```

Output from the job

Hands-on

Filling the empty space

- Note that in an iterative loop, the data must be exchanged between the real and the halo elements of the vector.
 - Find the right implementation in **DataTypes/vector.cpp**
 - Insert the call for the function into the right place
- Note that the L2-norm calculation should actually fire up an MPI call.
 - Which one?
 - Look at **MPI/common.cpp** to find the right implementation
- Note that **MPI/common.cpp** has a lot of functions marked with **NOT_IMPLEMENTED**. Replace this macro with correct MPI calls.

```
void Solver::copyVector(Vector &vec_in, Vector &vec_out) {  
    /*  
     * for every n-th elements in `vec_in`  
     *      assign element of vec_in(n) to vec_out(n)  
     */  
    NOT_IMPLEMENTED  
}  
  
void Solver::calculateResidual(Matrix &A, Vector &x,  
                               Vector &b, Vector &res) {  
    /*  
     * assign `b` to `res`  
     *  
     * for vector `x` and matrix `A`, the residual `res`  
     * is calculated as:  
     *      res(i) = b(i) - sum( A(i, j) * x(j) )  
     */  
    NOT_IMPLEMENTED  
}  
  
double Solver::calculateNorm(Vector &vec) {  
    /*  
     * for vector `vec` with n elements  
     *      L2-norm = sqrt( sum( vec(n) * vec(n) ) )  
     *                  |  
     *                  | local for every MPI process |  
     *                  |  
     *                  | one value for all MPI processes |  
     */  
    NOT_IMPLEMENTED  
    return 0.0;  
}
```

Tools

Tools

Overview: HPC tools

Tool name	Costs	Description
ARM DDT	Non-free	Full featured graphical, parallel debugger
HPCToolkit	Free	Integrated suite of tools for parallel program performance analysis
Intel One API	Free under certain conditions	Stack of different performance analysis and debugging tools (MPI/OpenMP/SIMD)
Valgrind	Free	Memory errors debugging tool
TotalView	Non-free	Full featured graphical, parallel debugger
Vampir	Non-free	Full featured trace visualizer for parallel program OTF trace files
memP	Free	Lightweight memory profiling tool
mpiP	Free	Lightweight MPI profiling tool
MUST	Free	MPI runtime error detection tool
PAPI	Free	A standardized and portable API for accessing performance counter hardware
likwid	Free	A tool to measure hardware performance counters
TAU	Free	Full featured parallel program performance analyses toolkit
Exrae	Free	MPI/OpenMP profiler
Scalasca	Free	performance analysis tool for MPI+OpenMP
Darshan	Free	IO profiler
nvprof	Free	Thread profiler (inc. GPU) from
gdb	Free	Standard GNU debugger
ARM MAP	Non-free	performance analysis tool for MPI+OpenMP
uProf	Free	performance analysis tool for MPI+OpenMP
gprof	Free	Standard unix/linux profiling utility

Different support for:

- Hardware
- Parallelisation strategies
- Compilers
- Interface

Other important aspects:

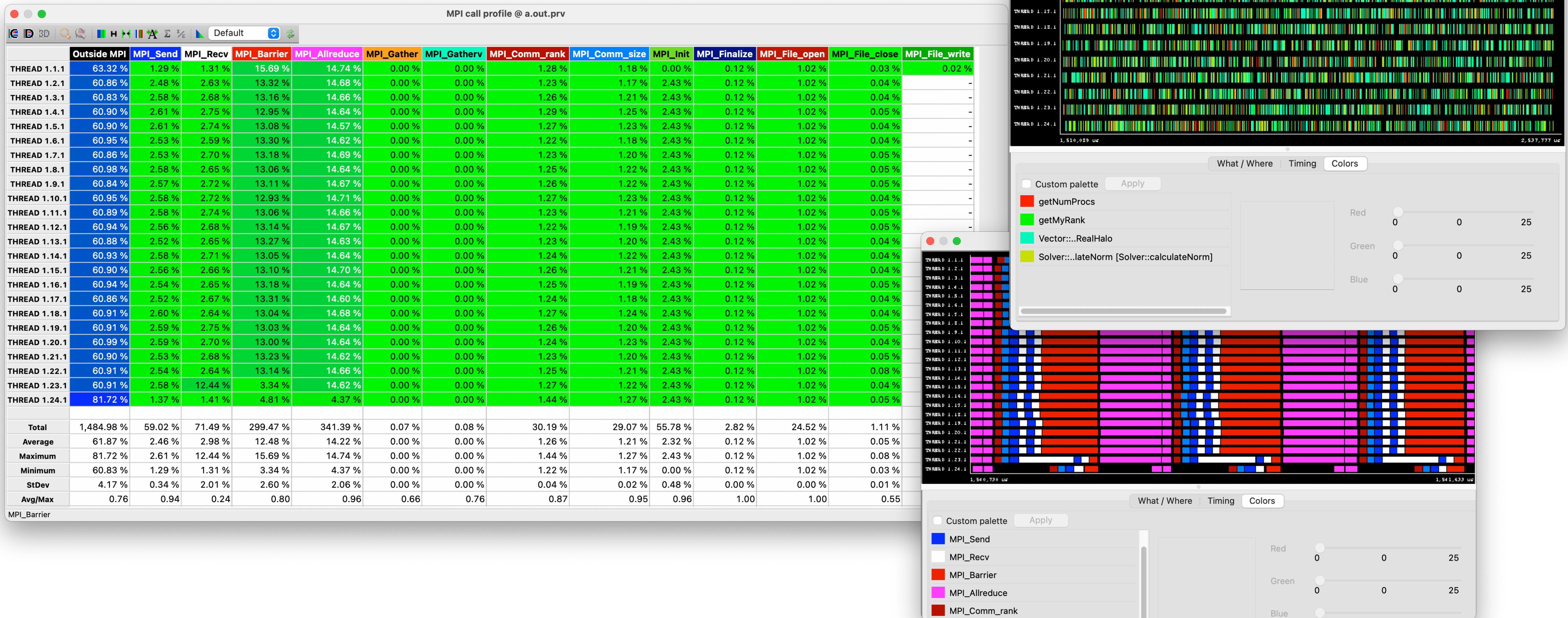
- Learning curve
- Completeness of the reports
- Costs and licenses
- Community support
- Documentation

Good overview:

<https://hpc.llnl.gov/software/development-environment-software>

Tools

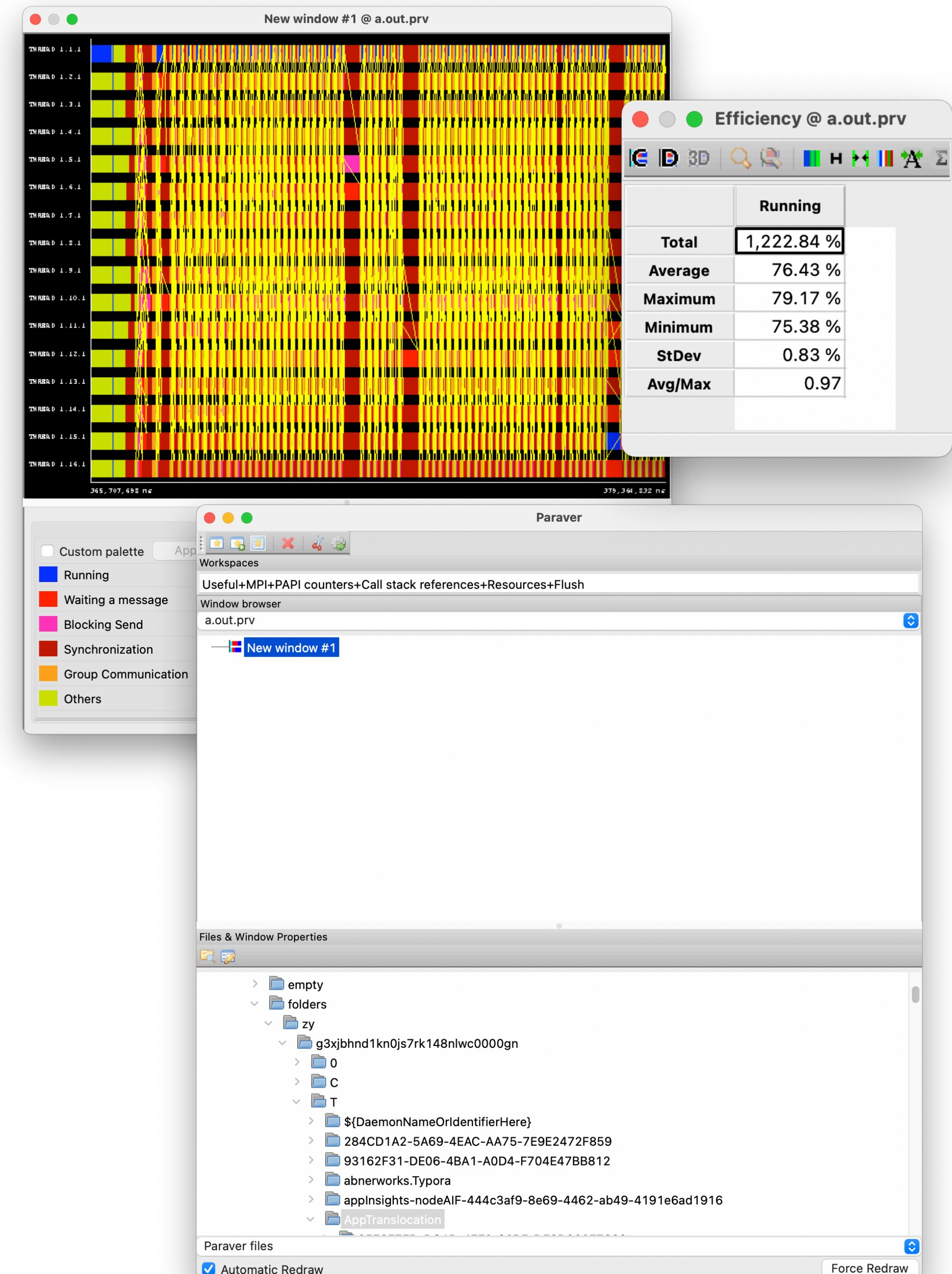
Overview: Extrae + Paraver



Hands-on

Hands-on Profiling

- Profile the code with **Exrae** (use 32x32 grid)
- Download **Paraver** to your local machine: <https://tools.bsc.es/downloads>
- Find the bottlenecks using **Paraver** (see `Hints` menu)
 - Visualise the “**MPI profile**”
 - Visualise the “**Profile caller line**”
 - Visualise the “**Caller function**”



Tools

Overview: Extrae + Paraver

```
$ module load 2020
$ module load Extrae/3.8.3-gompi-2020a
$ cd prace_jacobi
$ cp ${EBROOTEXTRAE}/share/example/MPI/ld-preload/trace.sh .
$ cp ${EBROOTEXTRAE}/share/example/MPI/extrae.xml .
$ sbatch test_extrae.sh
```

```
#!/bin/bash
#SBATCH --reservation=mpi_advanced
#SBATCH -n 16
#SBATCH --job-name=mpi_training
#SBATCH --output=out.p_np_16
#SBATCH --error=err.p_np_16
#SBATCH --time=0:05:00

module load 2020 foss/2020a
module load module load Extrae/3.8.3-gompi-2020a
### export trace.sh and executable
export EXE="./a.out -s 10 10 -d 4 4"
## Run the application
srun ./trace.sh $EXE
```

Use the following keys:

-s - set number of the grid cells in each direction (i j)
-d - set decomposition for each direction (i j)
(doesn't affect the METIS decomposition, but should be set anyway!)

```
#!/bin/bash

source /sw/arch/Debian10/EB_production/2020/software/
Extrae/3.8.3-gompi-2020a/etc/extrae.sh

export EXTRAE_CONFIG_FILE=./extrae.xml
# For C apps
export LD_PRELOAD=${EXTRAE_HOME}/lib/libmpitrace.so
# For Fortran apps
#export LD_PRELOAD=${EXTRAE_HOME}/lib/libmpitracef.so

## Run the desired program
$*
```