

HPC Exam Project

Scaling Study of the Stencil Method

Giovanni Lucarelli

October 19, 2025



UNIVERSITÀ
DEGLI STUDI
DITRIESTE

Introduction

Goals

1. **Optimize** the stencil method for the 2d heat equation
2. **Parallelize** using hybrid approach
3. Perform **scalability** study:
 - 3.1 Thread scaling
 - 3.2 Strong scaling
 - 3.3 Weak scaling

Algorithm

Heat equation (2d)

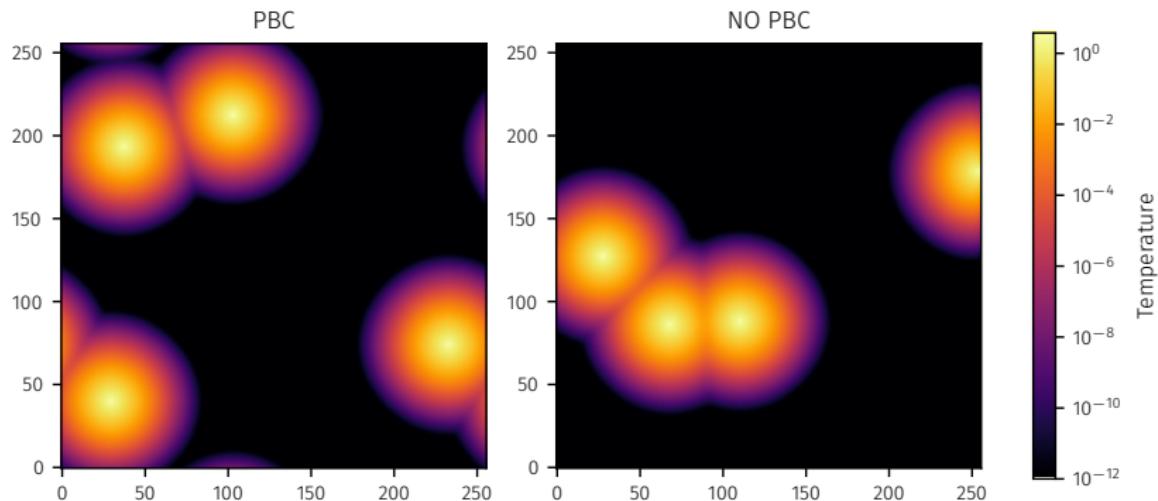
$$\partial_t u = \alpha(\partial_x^2 u + \partial_y^2 u)$$

Finite difference integration

$$u_{i,j}^{(t+1)} = (1 - 4\alpha)u_{i,j}^{(t)} + \alpha \sum_{\langle i,j \rangle} u_{i,j}^{(t)}$$

$$\begin{aligned} x \in [0, L_x] &\rightarrow i \in \{1, \dots, N_x - 1\} \\ y \in [0, L_y] &\rightarrow j \in \{1, \dots, N_y - 1\} \end{aligned}$$

Code Correctness



Optimization

- Compiler flags:

`-O3 -Wall -march=native`

- Preprocessor directive:

`#pragma GCC unroll`

Parallelization: shared memory

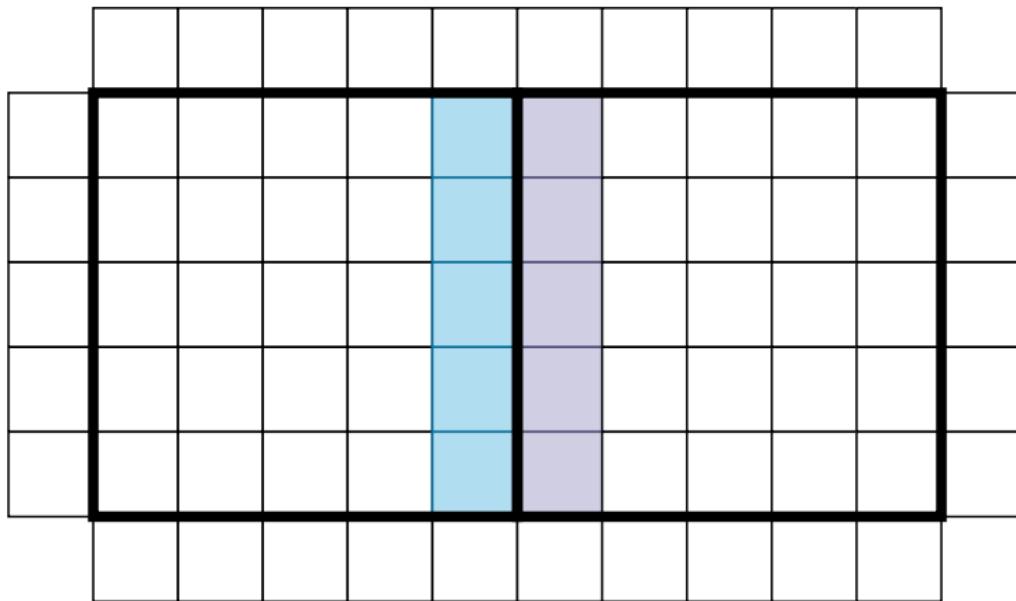
Implementation

```
1 #pragma omp parallel for schedule(static)
2 for (uint j = 1; j <= ysize; j++){
3     for ( uint i = 1; i <= xsize; i++){
4         // update rule
5     }
6 }
7 }
```

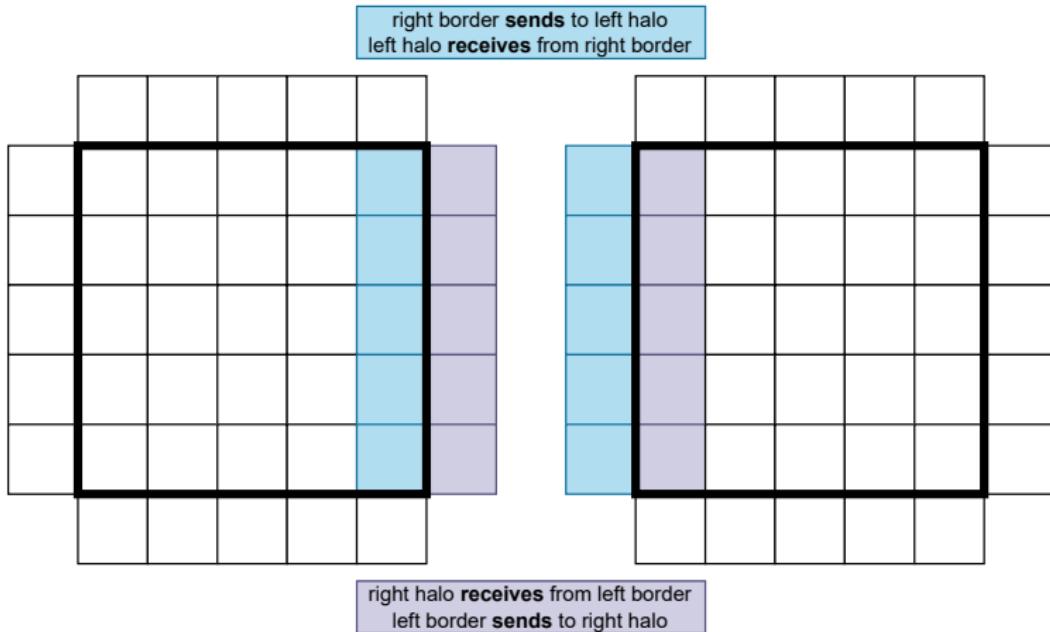
Thread placement and affinity

```
1 export OMP_PLACES=cores
2 export OMP_PROC_BIND=close
```

Parallelization: distributed memory



Parallelization: distributed memory



Parallelization: distributed memory

For each task:

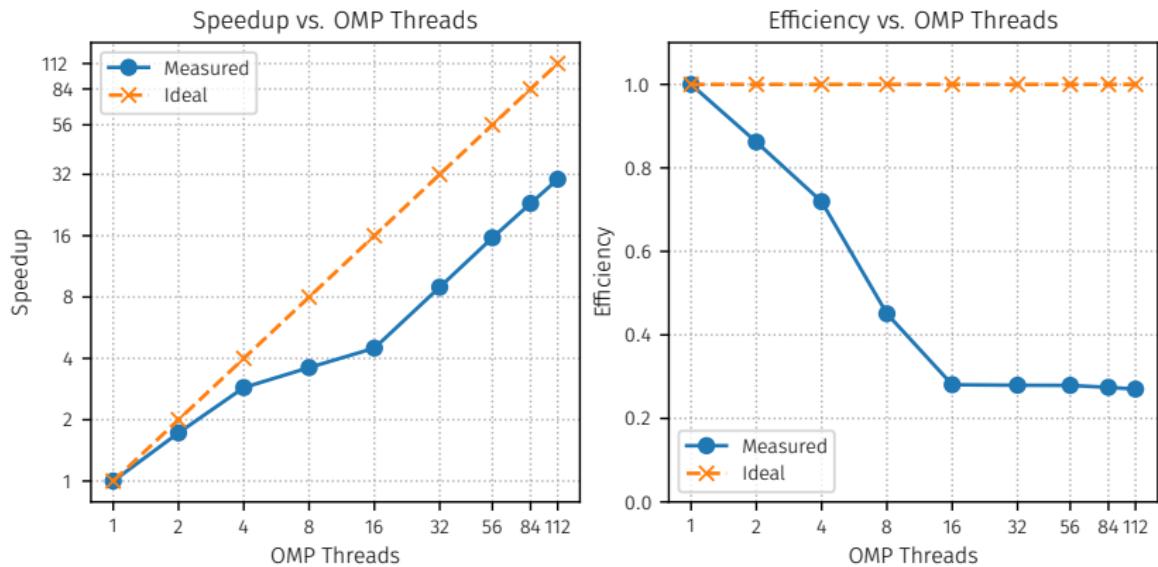
```
1      // pack buffers
2
3      MPI_Irecv(...);
4
5      MPI_Isend(...);
6
7      update_internal();
8
9      MPI_Waitall();
10
11     // unpack buffers
12
13     update_border();
```

Results

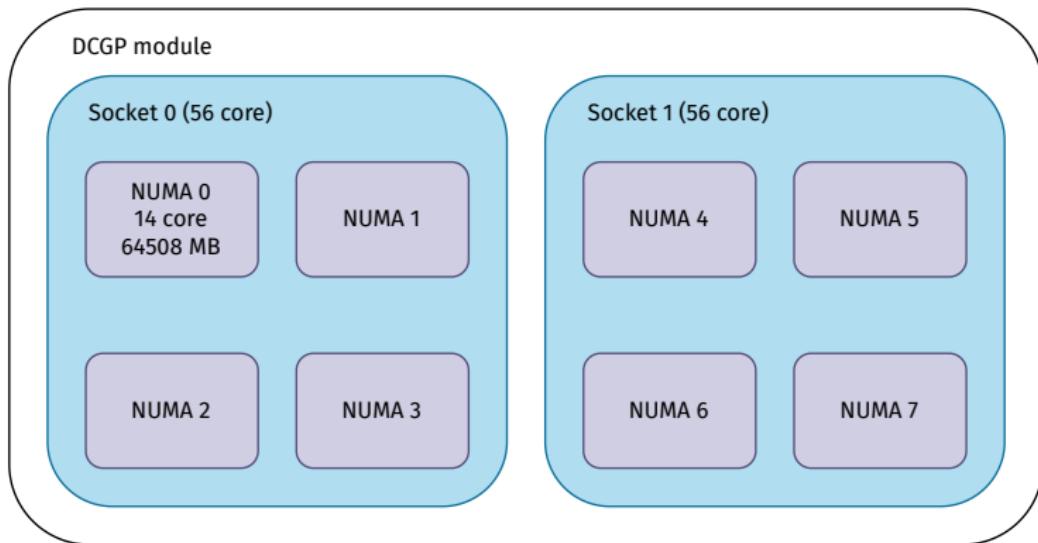
Thread Scaling

```
1      GRID_SIZE_X=16384
2      GRID_SIZE_Y=16384
3      N_STEPS=500
4
5      NODES=1
6      N_TASKS_PER_NODE=1
7
8      THREAD_LIST="1 2 4 8 16 32 56 84 112"
```

Thread Scaling



Node Architecture



Node Architecture: distance matrix

```
1 [glucarel@lrdn4293 HPC-leonardo]$ numactl --hardware
2     available: 8 nodes (0-7)
3
4     node   0   1   2   3   4   5   6   7
5       0:  10  12  12  12  21  21  21  21
6       1:  12  10  12  12  21  21  21  21
7       2:  12  12  10  12  21  21  21  21
8       3:  12  12  12  10  21  21  21  21
9       4:  21  21  21  21  10  12  12  12
10      5:  21  21  21  21  12  10  12  12
11      6:  21  21  21  21  12  12  10  12
12      7:  21  21  21  21  12  12  12  10
```

Node Architecture: memory

```
1 [glucarel@lrdn4293 HPC-leonardo]$ lscpu | egrep 'L1d|L1i|L2|L3'  
2     L1d cache:          48K  
3     L1i cache:          32K  
4     L2 cache:         2048K  
5     L3 cache:        107520K
```

$$N = N_x = N_y = 2^{14}$$

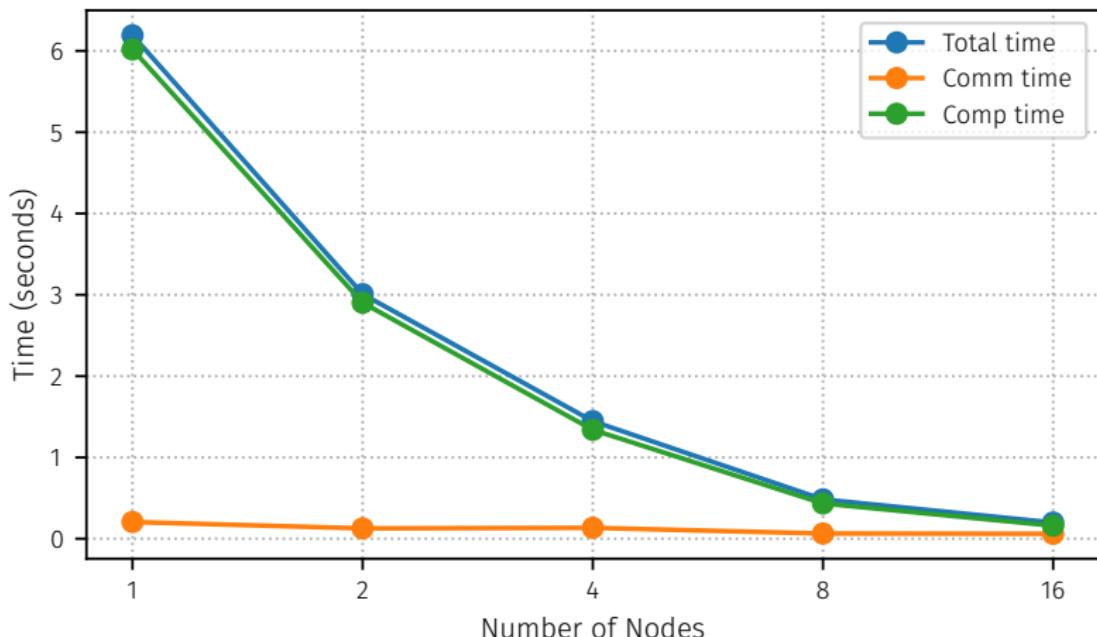
$$\text{memory}_{grid} = 2 \times N^2 \times 8B \approx 4\text{GB}$$

Strong Scaling

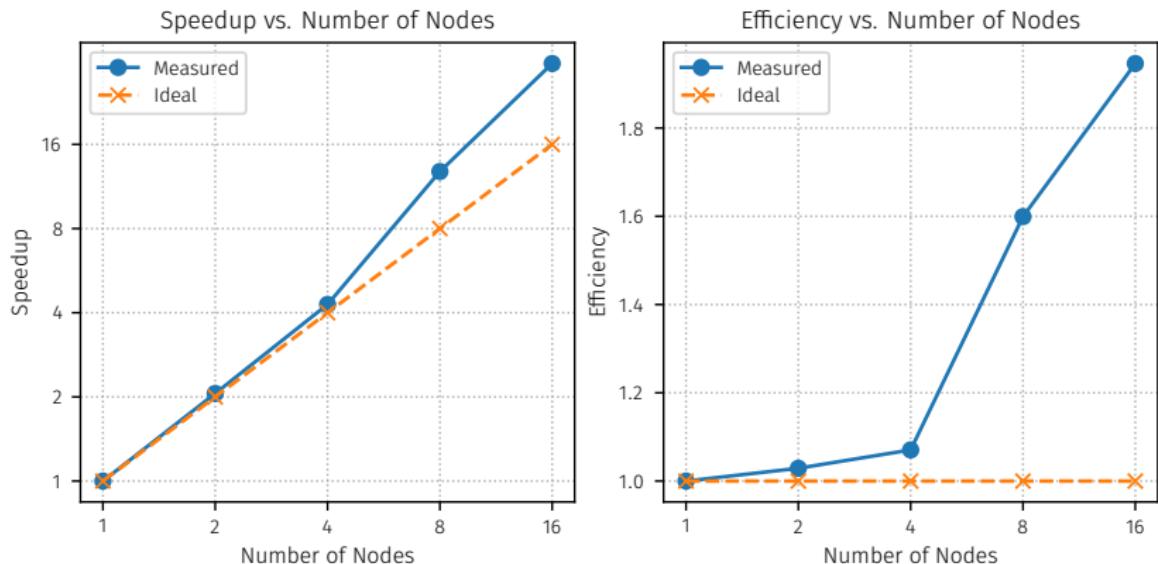
```
1      GRID_SIZE_X=16384
2      GRID_SIZE_Y=16384
3      N_STEPS=500
4
5      OMP_THREADS=14
6      N_TASKS_PER_NODE=8
7
8      NODE_LIST="1 2 4 8 16"
```

Strong Scaling

Time Components vs. Number of Nodes (Strong Scaling)



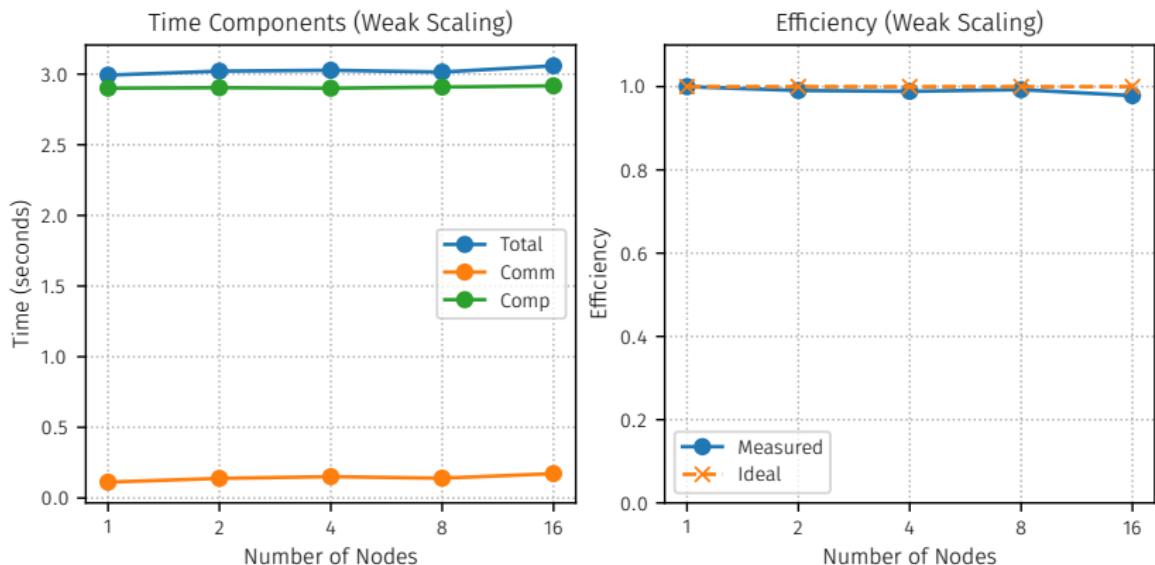
Strong Scaling



Weak Scaling

```
1      LOCAL_X=4096
2      LOCAL_Y=4096
3
4      OMP_THREADS=14
5      TASKS_PER_NODE=8
6
7      for NODES in "1 2 4 8 16"; do
8          TOTAL_TASKS=$(( NODES * TASKS_PER_NODE ))
9
10         case "${TOTAL_TASKS}" in
11             8)      PX=4;   PY=2   ;;    # 1 node  (8 ranks)
12             ...
13         esac
14
15         GRID_SIZE_X=$(( LOCAL_X * PX ))
16         GRID_SIZE_Y=$(( LOCAL_Y * PY ))
17         ...
18     done
```

Weak Scaling



Conclusion

- Stencil method is memory bound!

Thank You!