Session 2: The PIC method
and its parallelization

Smilei Workshop

Parallel computing

9 November 2023

Francesco Massimo (z10f )
francesco.massimo@universite-paris-saclay.fr

Outline

- What is parallel computing?
- Basic Supercomputer Architecture
- Splitting your simulation domain
- MPI+OpenMP parallelization
- Balancing the load between OpenMP threads
- Balancing the load between MPI processes
- F.A.Q: how to setup a simulation?

What is parallel computing?

Example: serial program

1 Computing unit = 1 node or 1 socket or 1 GPU, etc

Computing unit 1:

A=A+1 (10 s)
B=B*2 (10 s)
C=C/3 (15 s)
D=D*D (10 s)

Total time to execute the program: 45 s

Example of (almost) balanced program: Using 4 computing units the execution time is reduced

Computing unit 1

A=A+1 (10 s)

Total computing time: 10 s

Computing unit 2

B=B*2 (10 s)

Total computing time: 10 s

Computing unit 3

C=C/3 (15 s)

Total computing time: 15 s

Computing unit 4

D=D*D (10 s)

Total computing time: 10 s

Total time to execute the program: 15s

The total execution time is determined by the slowest computing unit

Computing unit 1

A=A+1 (10 s)

Total computing time: 10 s

Computing unit 2

B=B*2 (10 s)

Total computing time: 10 s

Computing unit 3

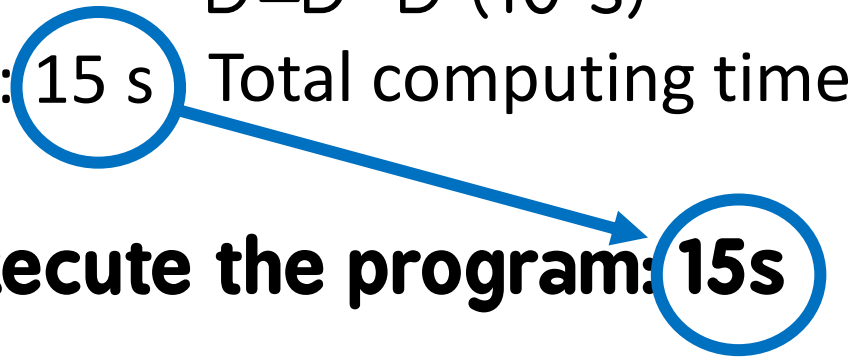
C=C/3 (15 s)

Total computing time: 15 s

Computing unit 4

D=D*D (10 s)

Total computing time: 10 s



Total time to execute the program: 15s

Extreme example of program with unbalanced load

Computing unit 1
Total core time: 1 s

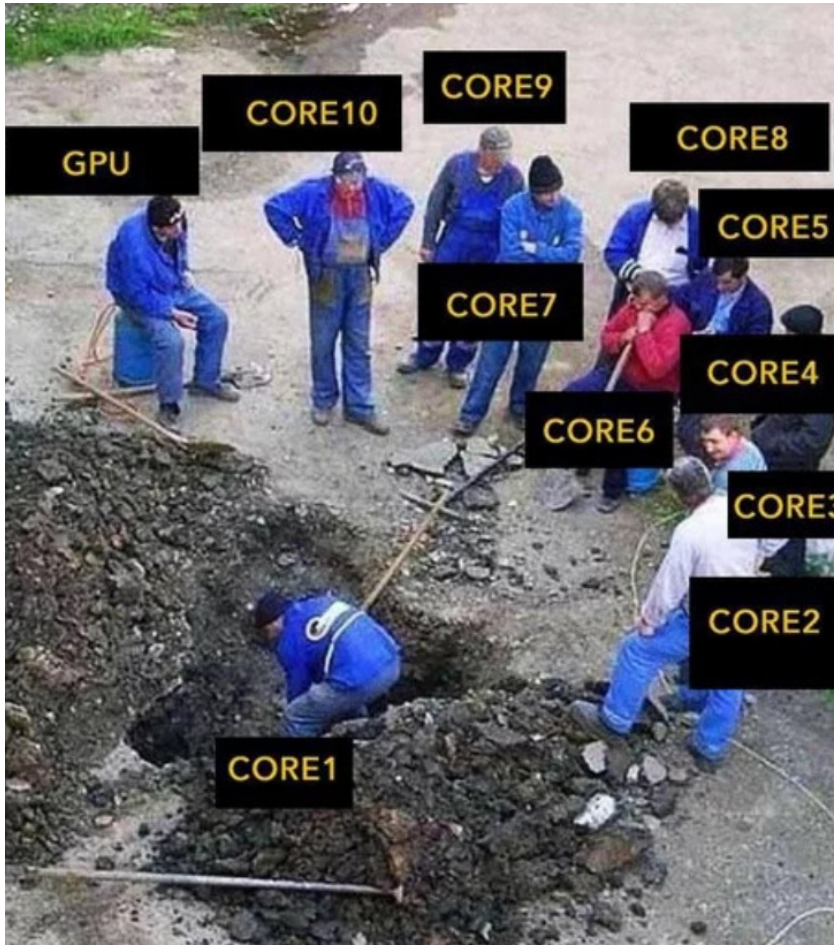
Computing unit 2
Total core time: 2 s

Computing unit 3
Total core time: 5 s

Computing unit 4
Total core time: 1000 s

Total time to execute the serial program: 1008s
Total time to execute the parallel program: 1000s

Conclusion: keep load balance between computing units



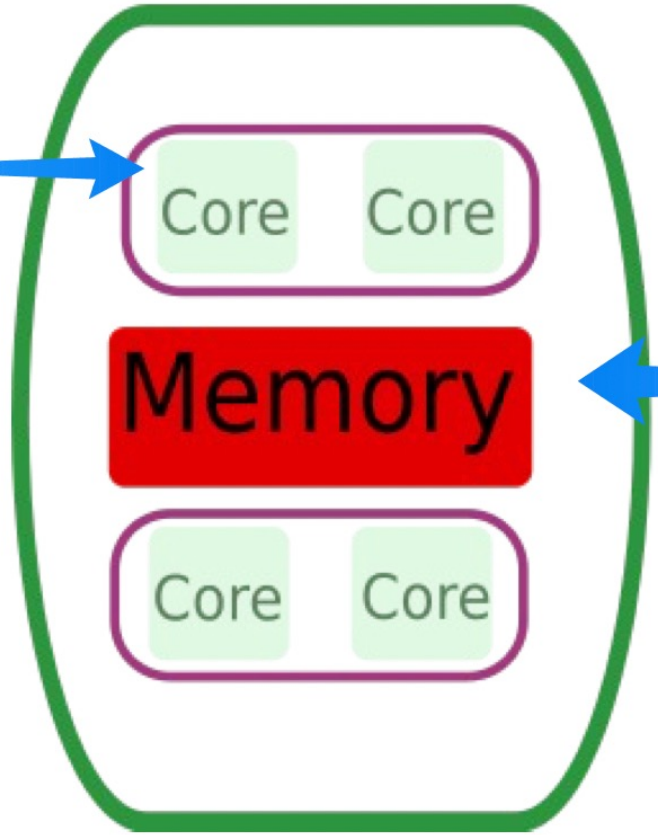
It's totally useless
to use more parallel computing units
if only one or a few
are doing all the work

Decompose your program in small
parallelizable units and
distribute them evenly
between computing units
working in parallel

Basic Supercomputer Architecture

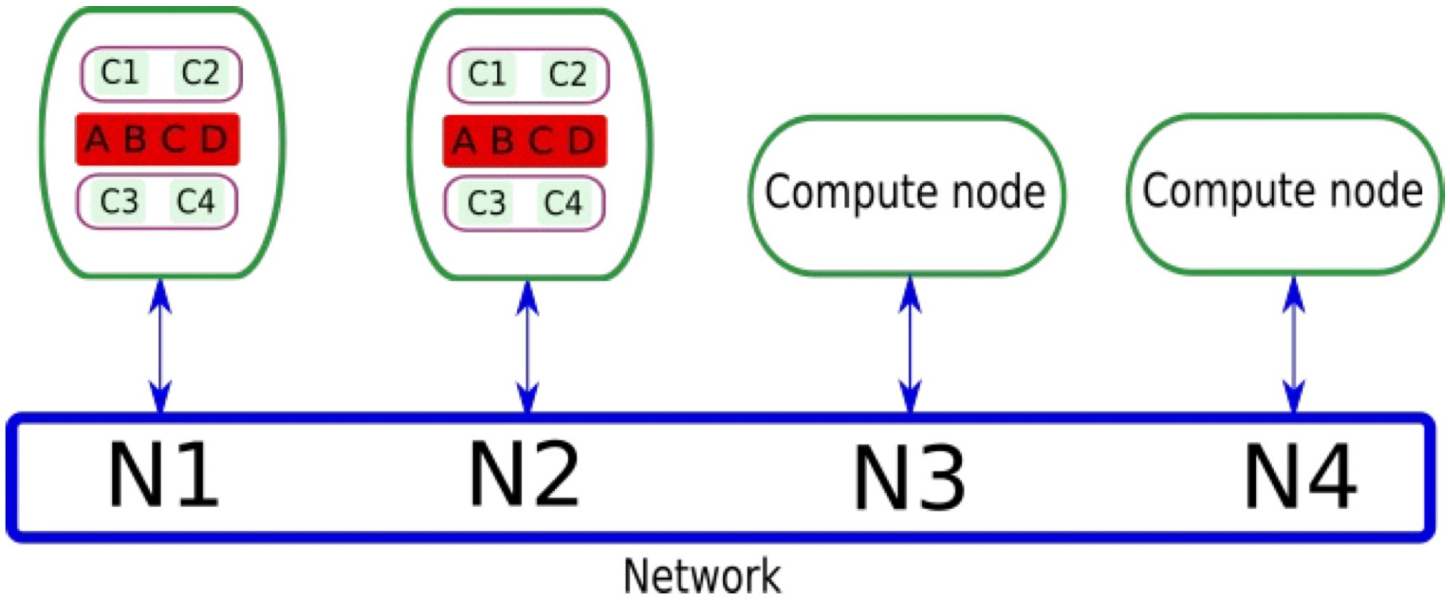
Example of computing unit: compute node with shared memory system

4 cores
for computation



Shared
memory
for data

Example: distributed machine made of many compute nodes, each with shared memory system



Communications in Smilei

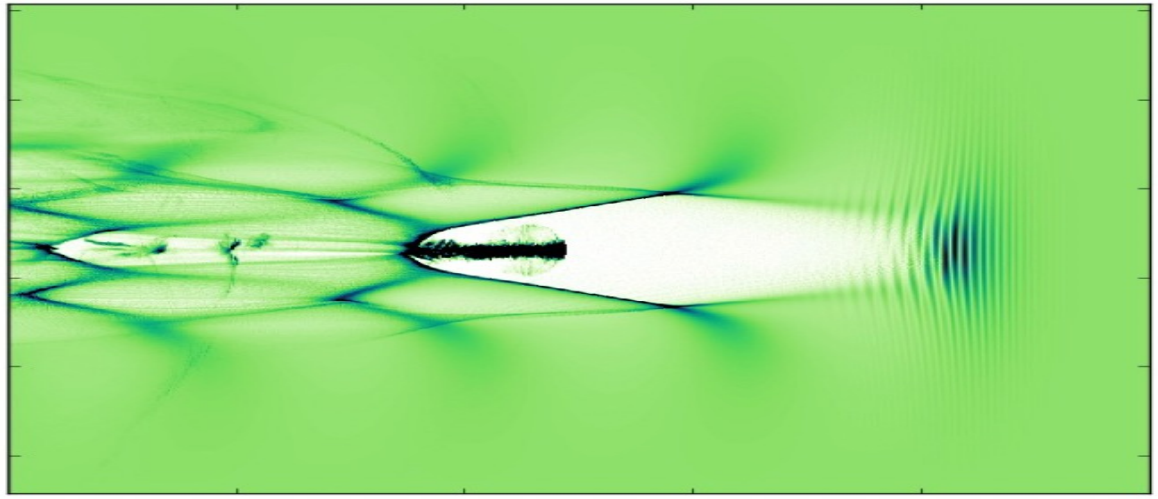
- Intranode communications:
  → **OpenMP**

- Internode communications:
  → **MPI**

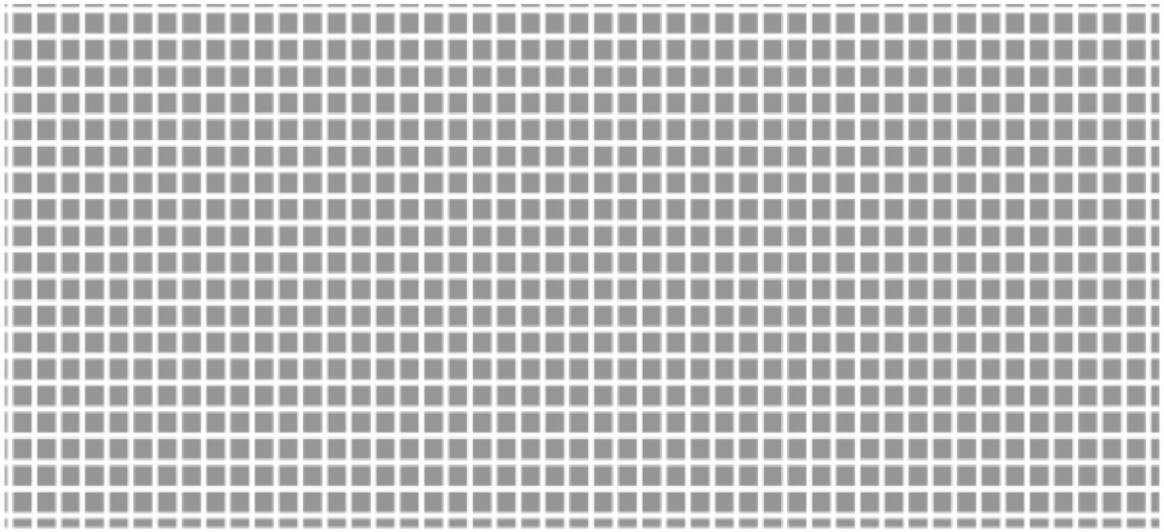
Running a Smilei simulation on a distributed machine,
- how do we distribute the operations?
- How do we keep load balance?

Splitting your simulation domain

Remember: a PIC discretizes space with cells

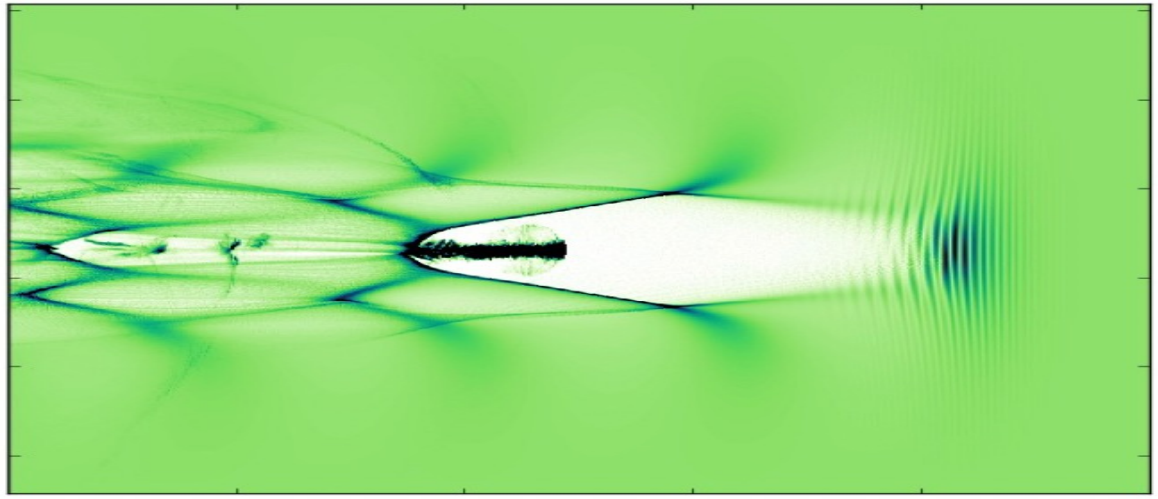


What you see:
Laser Wakefield Acceleration

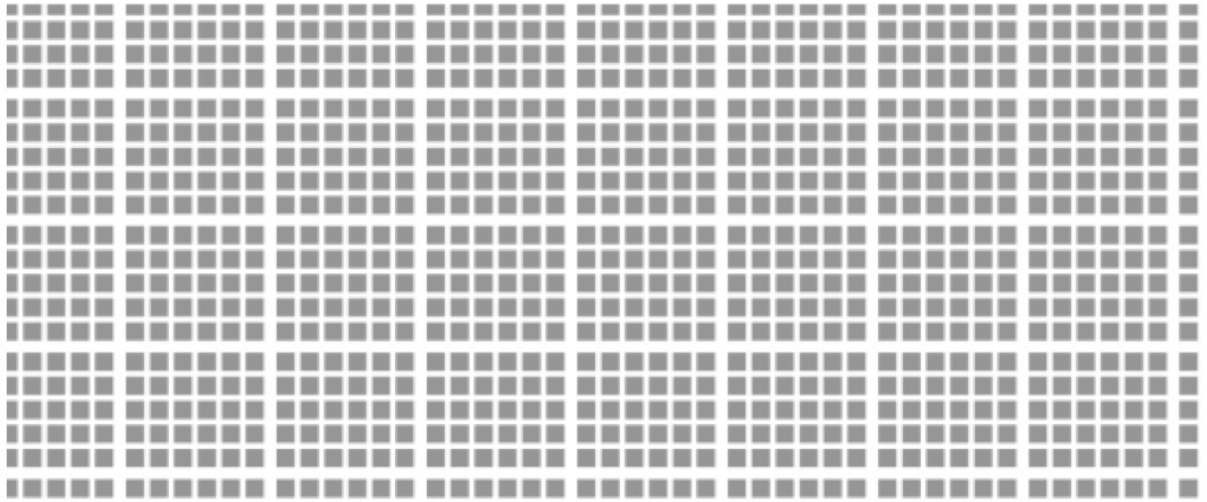


What the computer sees:
a collection of **cells** (figure not in scale)
populated by fields and macro-particles

In Smilei, cells are grouped in patches



What you see:
Laser Wakefield Acceleration



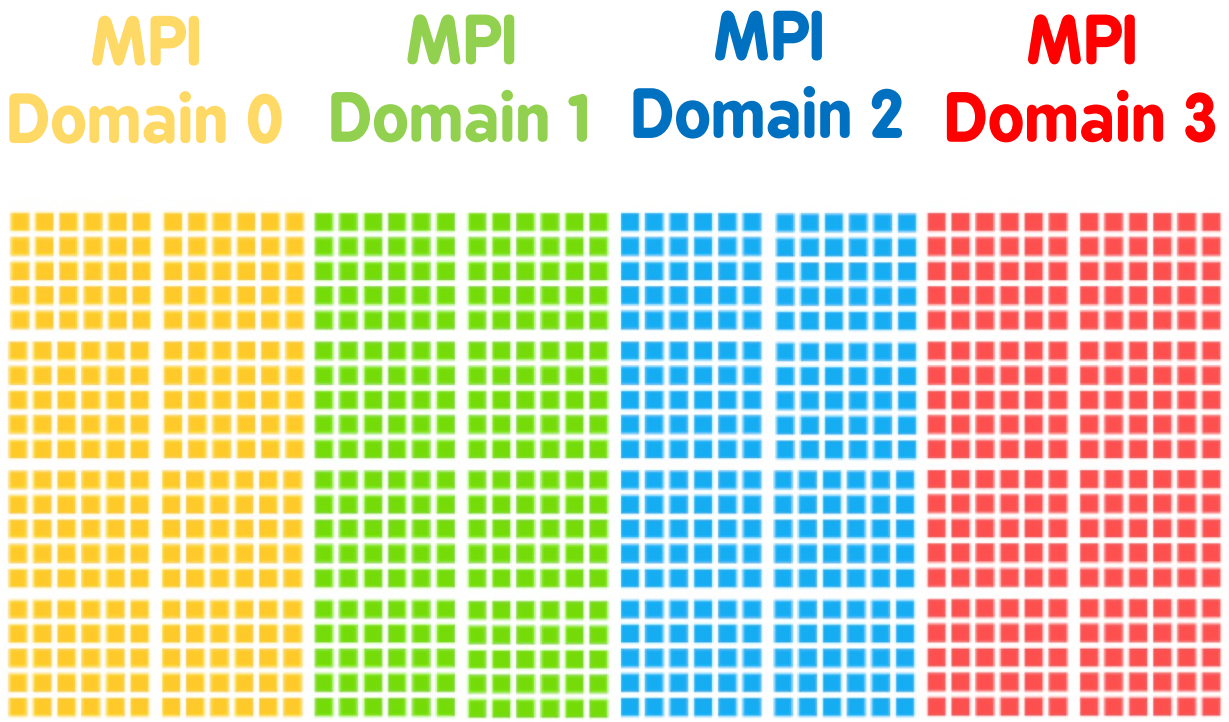
What the computer sees:
a collection **patches** made of cells
(figure not in scale)
populated by fields and macro-particles

MPI+OpenMP parallelization

In Smilei, patches are grouped in different memory locations = MPI domains

MPI domains are made of contiguous patches

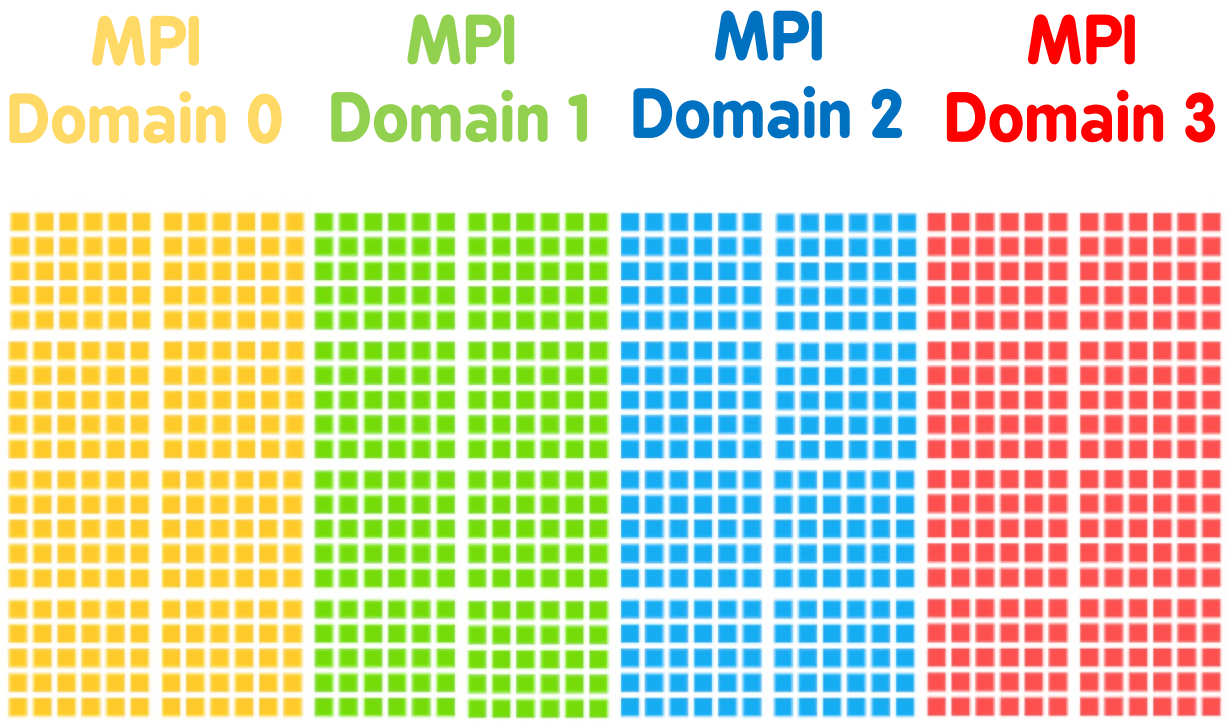
Example of Cartesian MPI decomposition (possible in Smilei but not default)



In Smilei, 1 MPI process handles 1 MPI domain

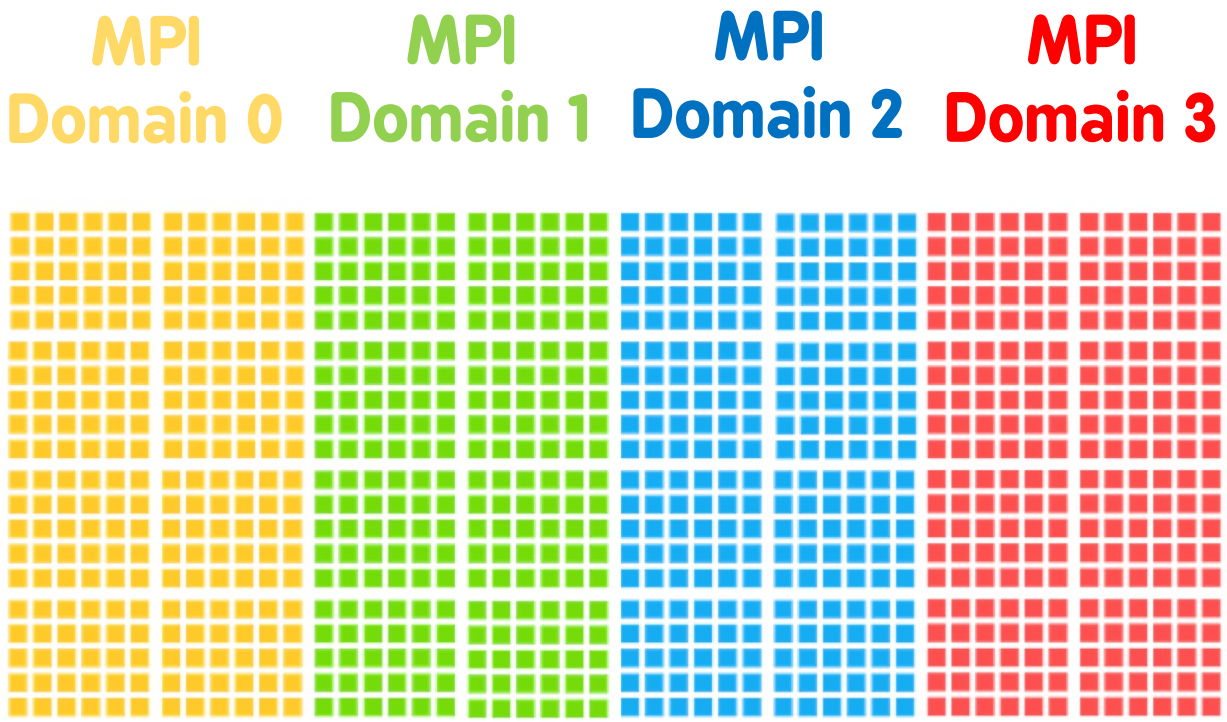
1 MPI process includes
a fixed number of OpenMP threads

Total computing cores
= # MPI processes x # OpenMPthreads



Note: 1 computing node can have more than 1 MPI process

MPI domain decomposition → need for MPI communications

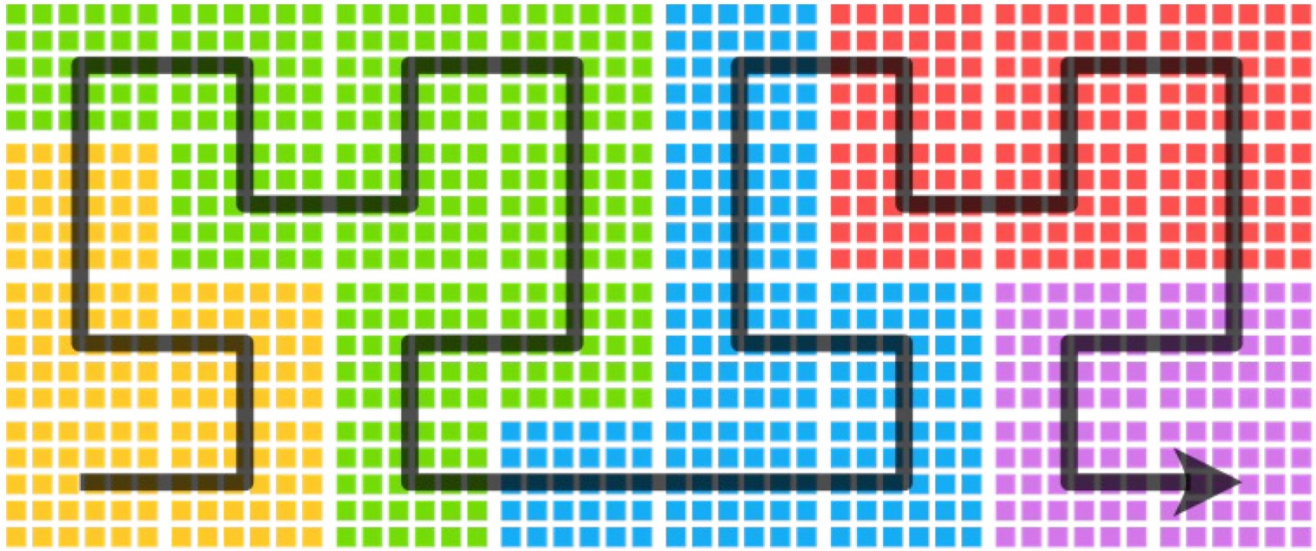


OpenMP threads share the memory
MPI processes do not share the memory

⟶ MPI domains must communicate with each other at their borders

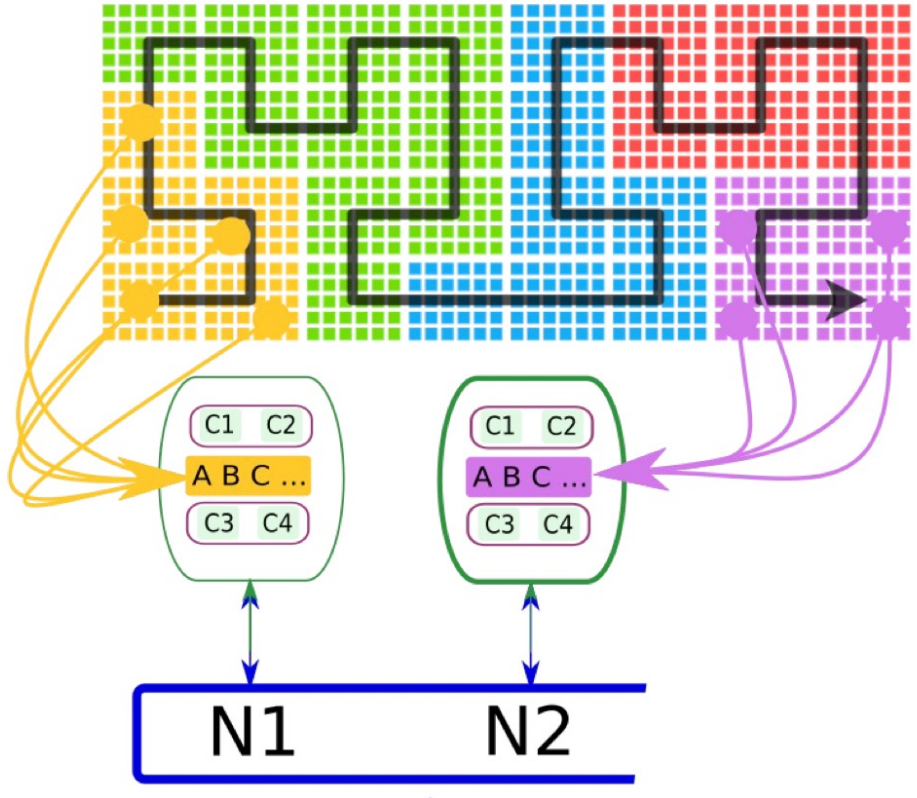
More common MPI decomposition in Smilei

Hilbertian MPI decomposition
(good for patch exchange)



MPI Domain 0 | MPI Domain 1 | MPI Domain 2 | MPI Domain 3 | MPI Domain 4

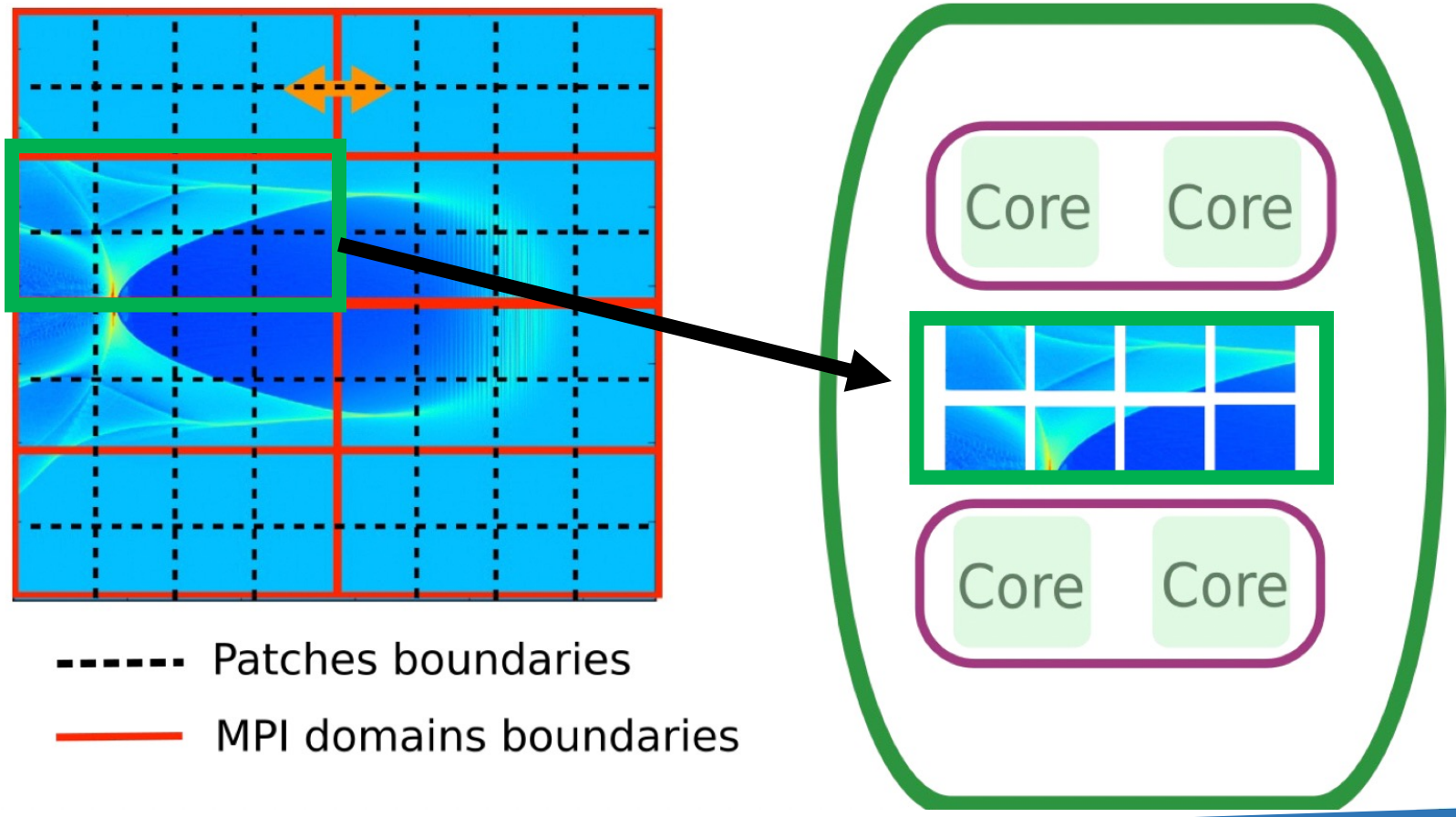
MPI domains are assigned to computing nodes



MPI Domain 1 MPI Domain 2 MPI Domain 3 MPI Domain 4 MPI Domain 5

Ok, but where are the patches in the supercomputer?

All patches of the MPI domain owned by the local node are stored in the memory



How do we keep the everything balanced?

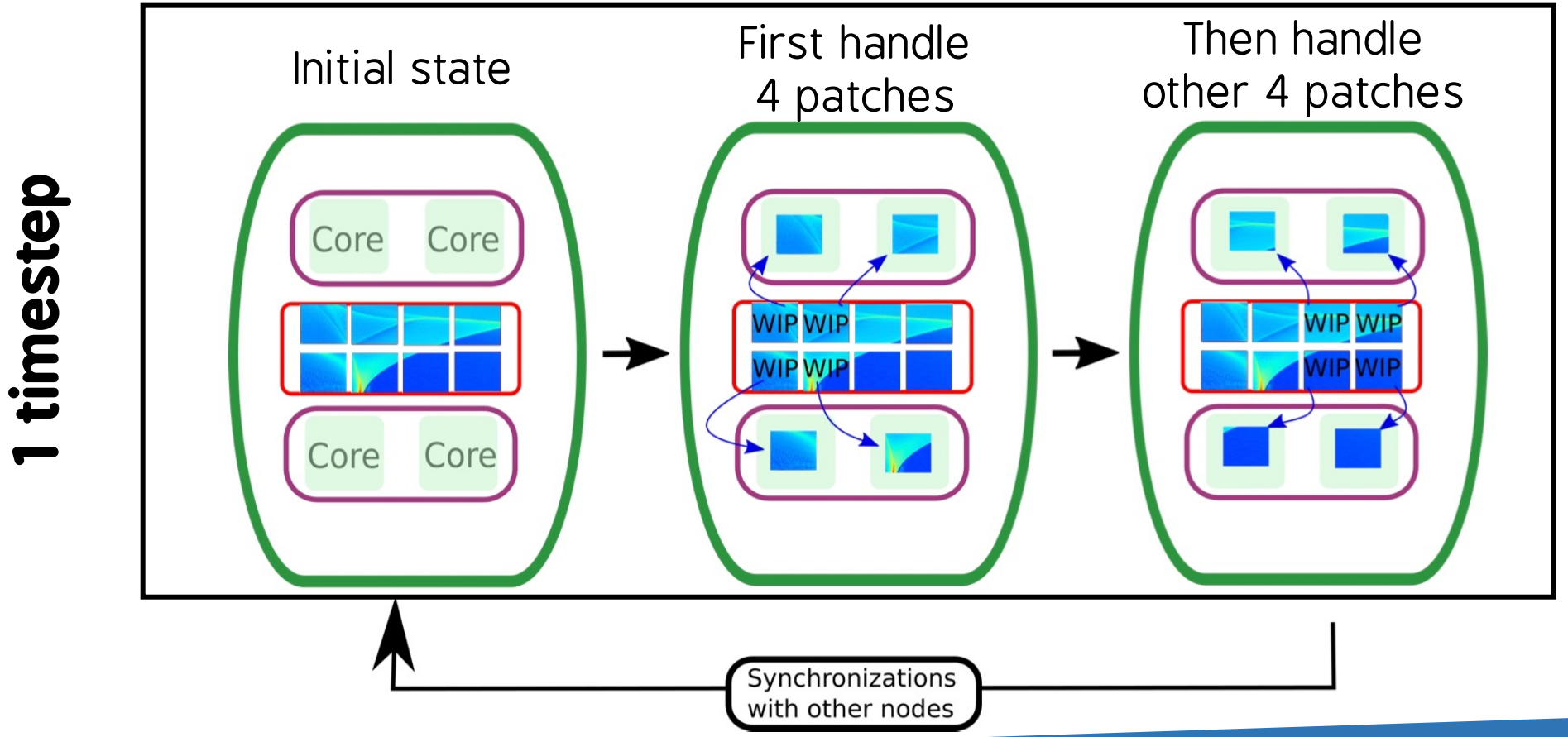
- between OpenMP threads? ⟶ **OpenMP dynamic scheduling**

- between MPI processes? ⟶ **Dynamic MPI load balancing**

Balancing the load between OpenMP threads

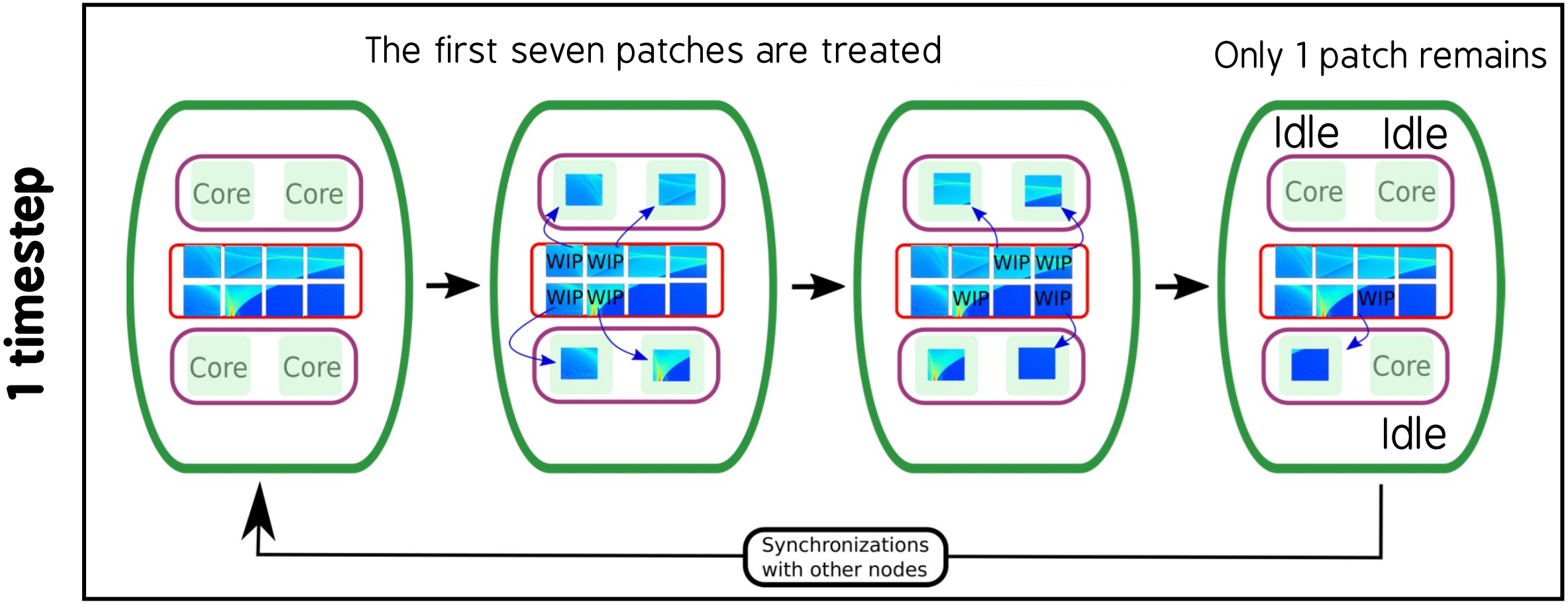
The OpenMP scheduler distributes patches to threads

The OpenMP scheduler assigns cores to patches via the **openMP threads**.
The number of OpenMP threads is fixed by the user and should be **one per core**.



OpenMP threads and load imbalance

Imbalance of patch loads induces idle time

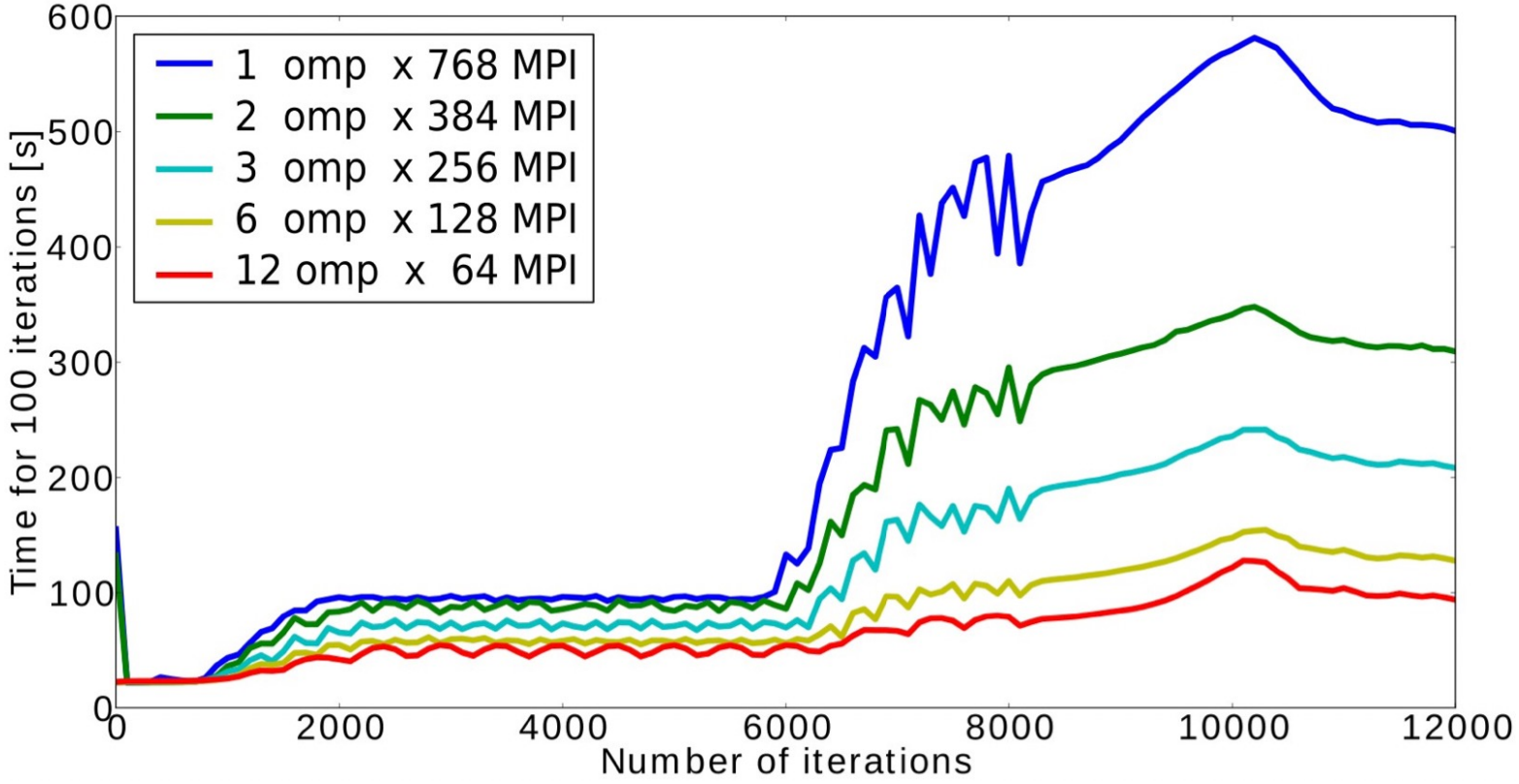


The OpenMP dynamic scheduler balances the load

- Generally, number of threads = number of cores (no hyper threading)
- More threads, better balance only if many more patches than threads in order to hopefully average the load.

The OpenMP dynamic scheduler balances the load

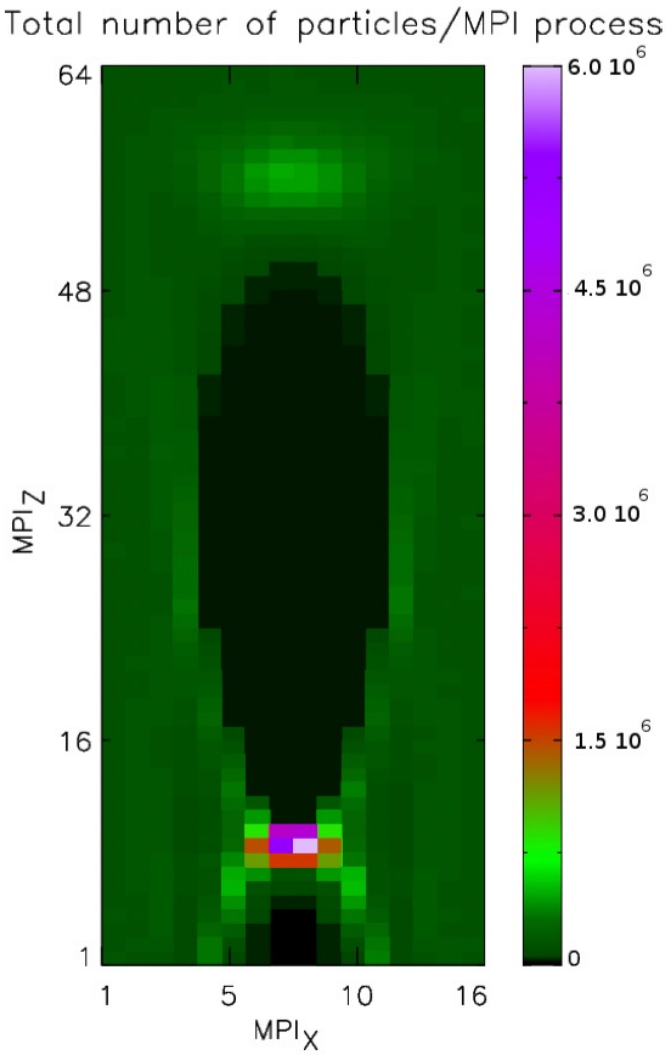
Example with increasing load imbalance:
Laser Wakefield Acceleration



Note: no MPI load balancing is used for this figure!
(See following slides)

Balancing the load between MPI processes

Load imbalance also occurs at the MPI level → between computing nodes



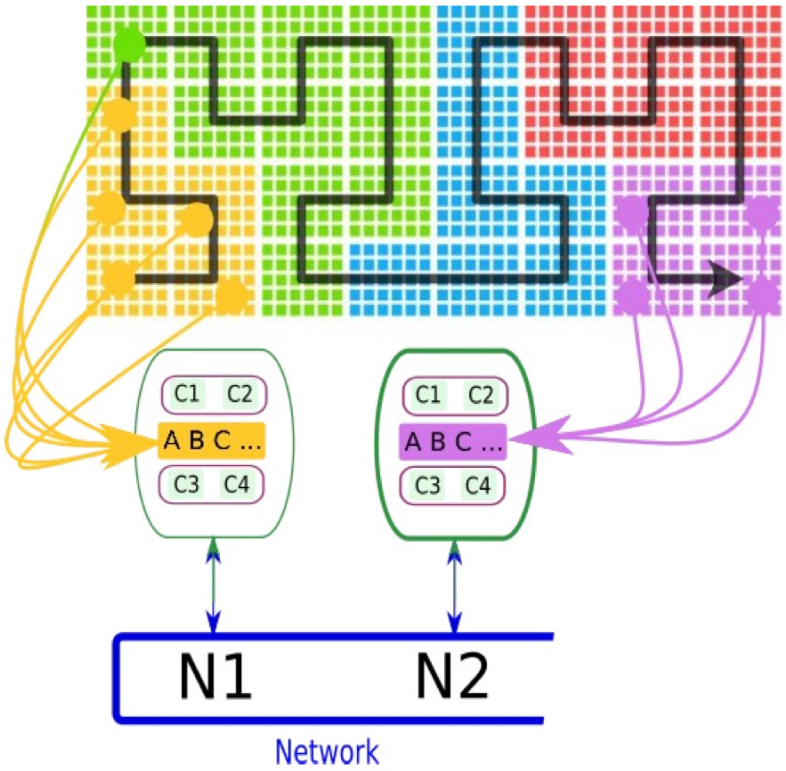
At first order,
total number of macro-particles → computing load

Macro-particles can change MPI domain
and their number in a MPI domain can evolve in time.

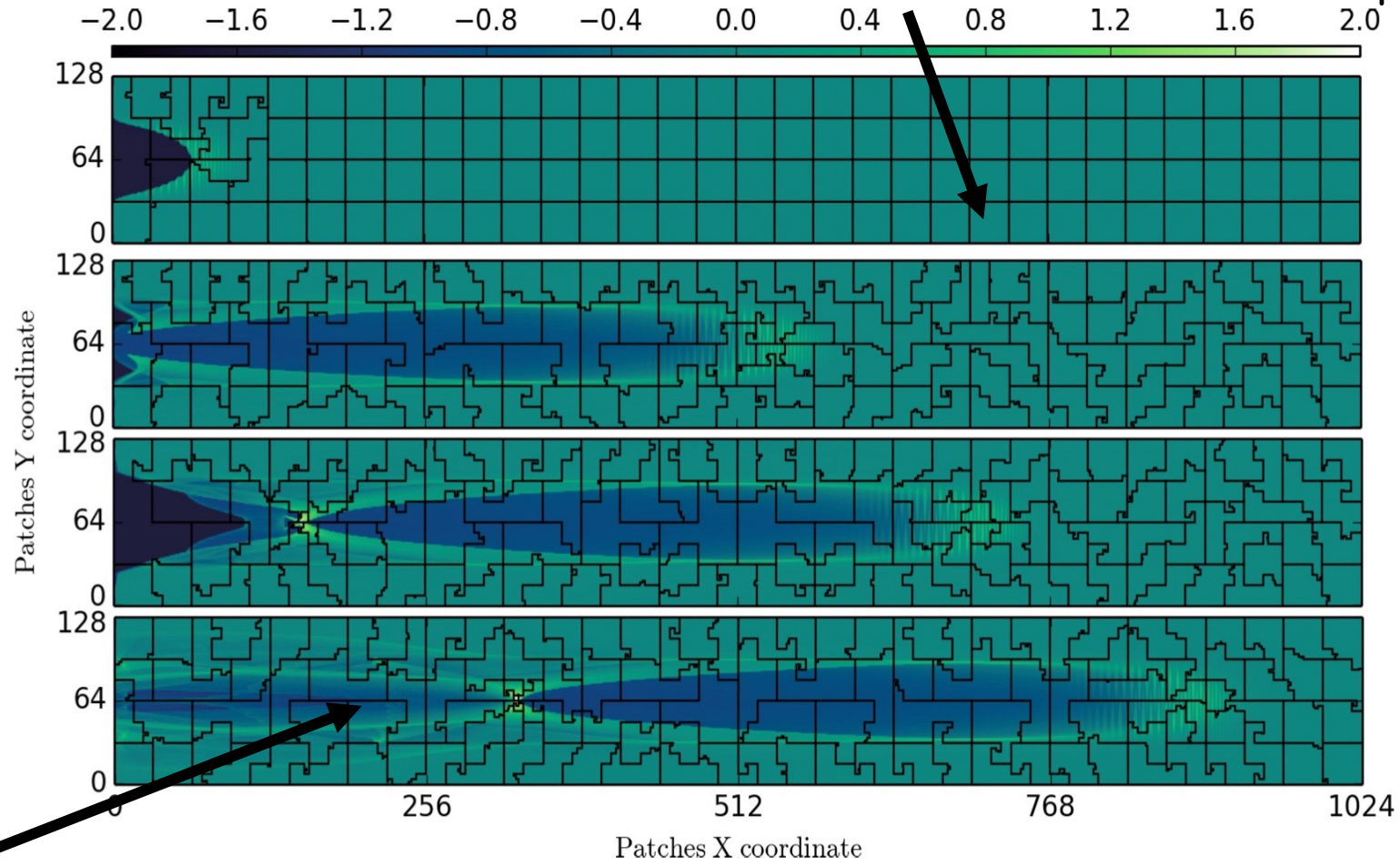
To maintain balance between MPI domains, we must change the number of macro-particles in a MPI domain

The number of macro-particles in a MPI domain can be done through patch exchange

Patches are exchanged between MPI domains through dynamic load balancing between MPI processes



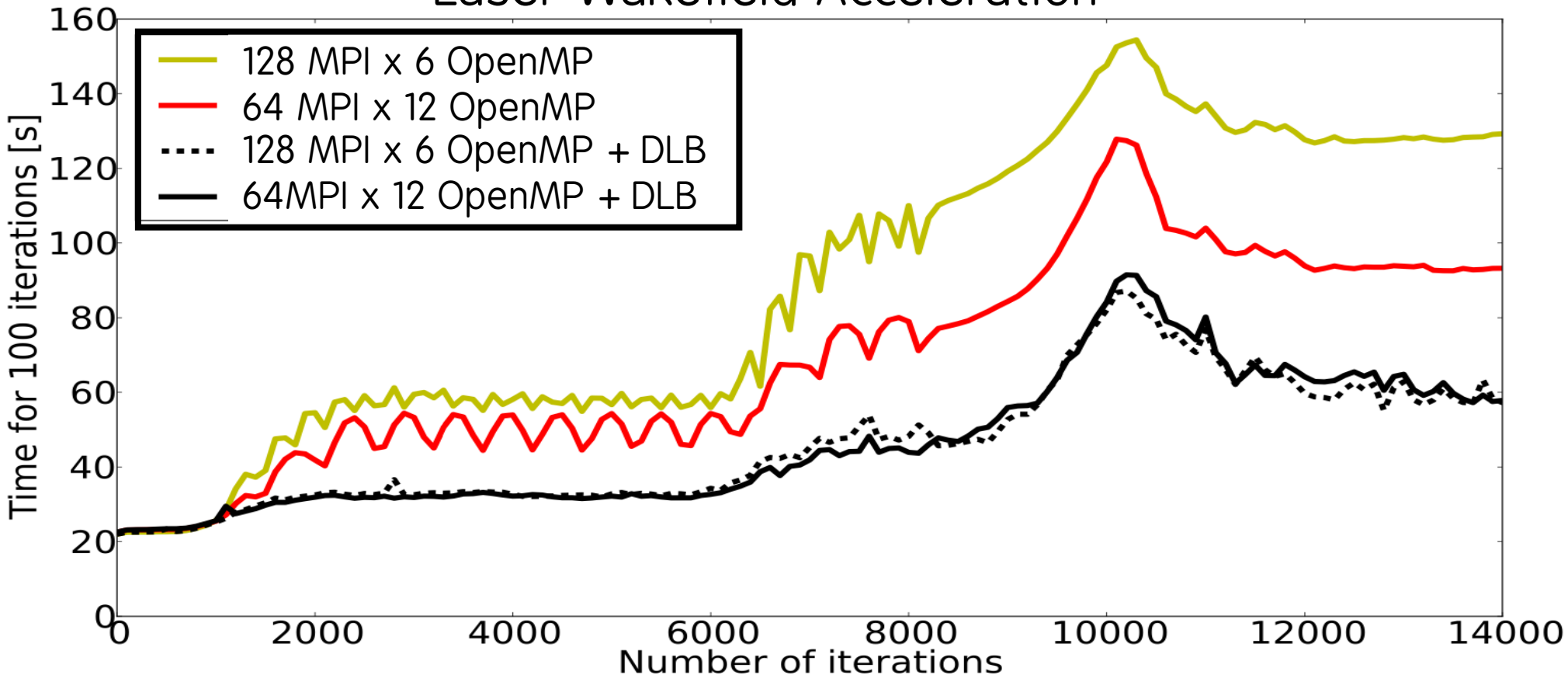
MPI domains still with their initial shape



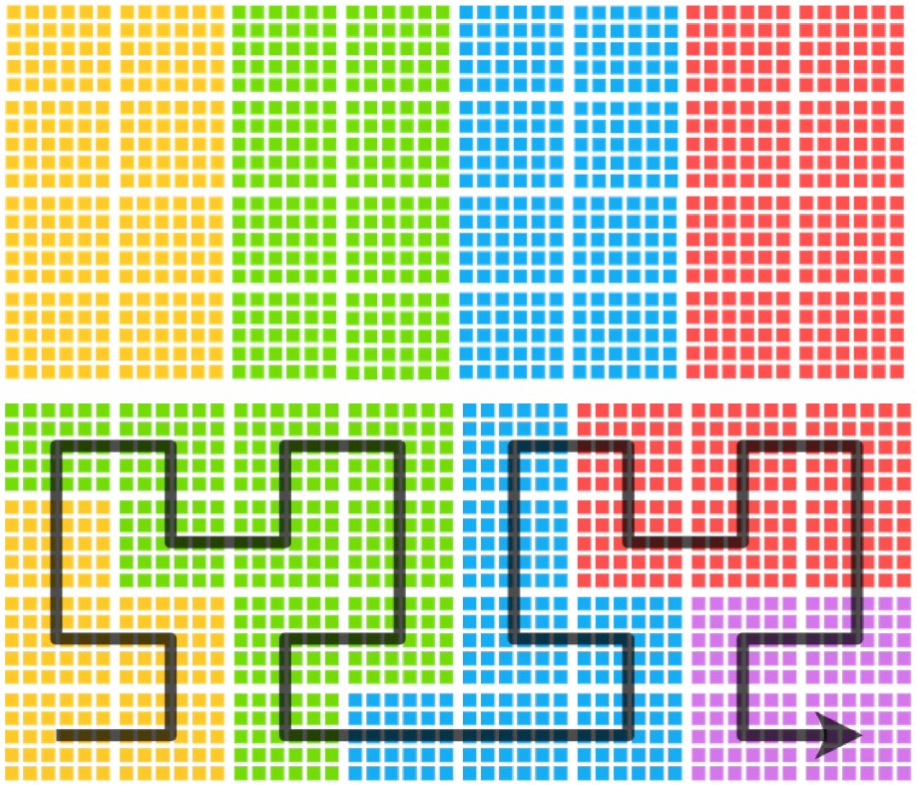
MPI domains with
Irregular shape
after patch exchanges

Effects of Dynamic Load Balancing (DLB) between MPI

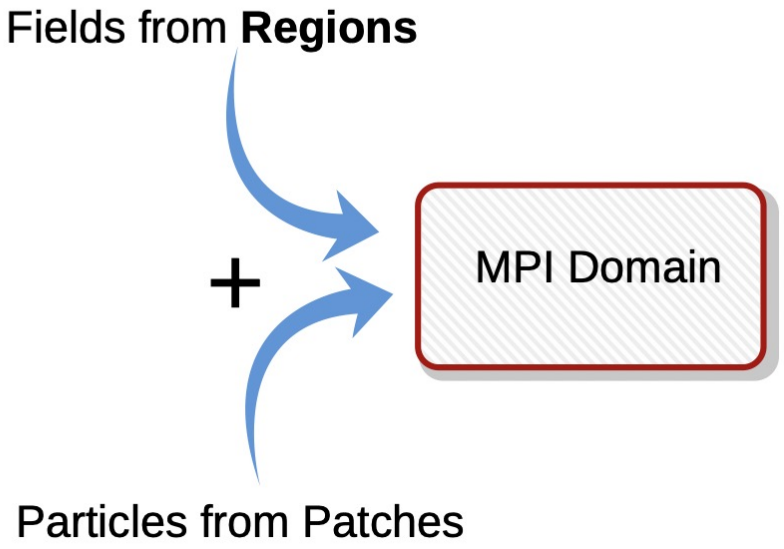
Example with increasing load imbalance:
Laser Wakefield Acceleration



Smilei feature:
Single Domain, Multiple Decompositions (SDMD)



J. Derouillat and A. Beck,
Phys.: Conf. Ser. 1596, 012052 (2020)



Use small patches without heavy synchronization costs
Very useful for heavy operation on fields e.g. current filters

Glossary

Hardware	Software	Associated Data structure
Node	MPI process	Group of Patches
Core	OpenMP thread	1 Patch

F.A.Q: how to setup a simulation?

Which MPI+OpenMP set-up should I use?

- 1 or few MPI per socket (usually 2 per node)
- 1 OpenMP thread per core in the socket:

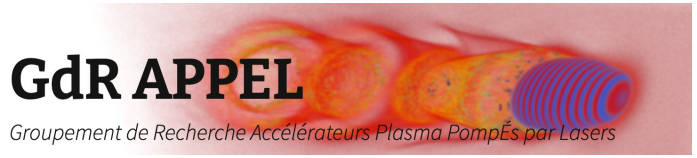
```
export OMP_NUM_THREADS = …
```

- Use OpenMP dynamic scheduler if your case is imbalanced:

```
export OMP_SCHEDULE=dynamic
```

Thank you for your attention!

Thanks for supporting this event



Contributing labs, institutions & funding agencies