Smilei Workshop

March 2022

Parallelization

Arnaud Beck

Laboratoire Leprince-Ringuet, CNRS / Ecole polytechnique

Outline

I. Basic Supercomputer Architecture

II. The Domain Decomposition Method

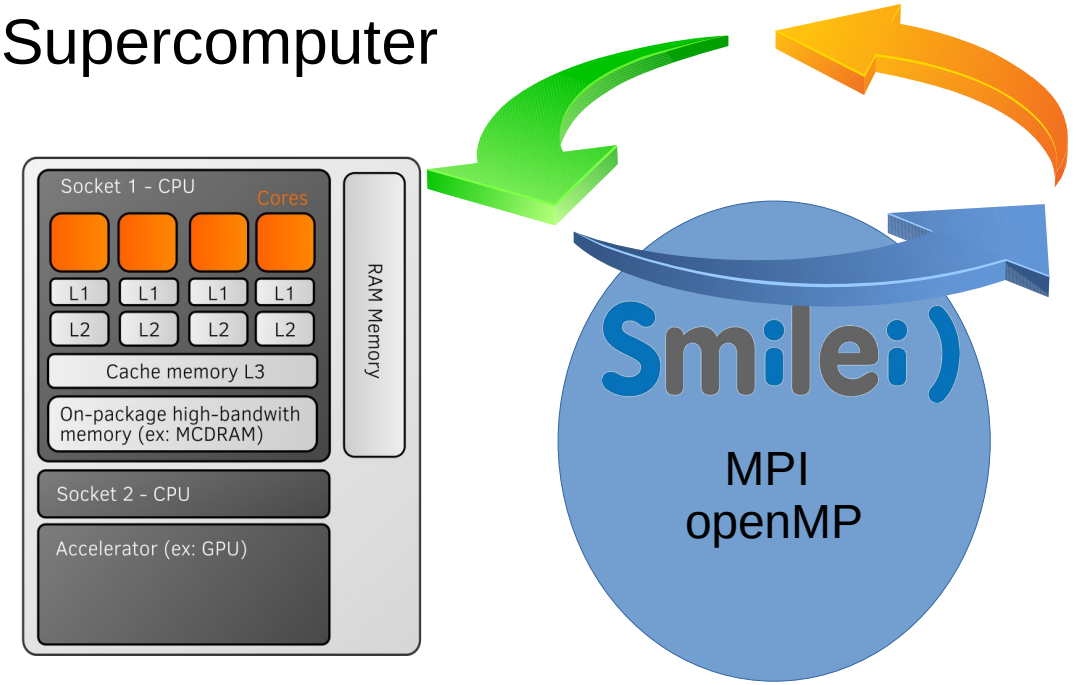
III. MPI+openMP setup

III. Dynamic Load Balancing

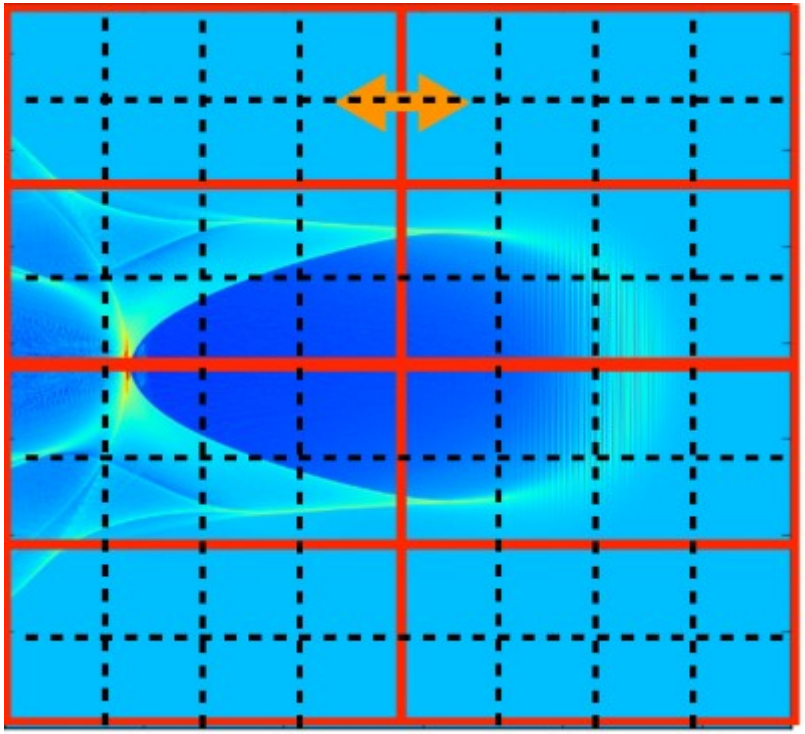
IV. F.A.Q about how to setup a simulation

Outline

Supercomputer

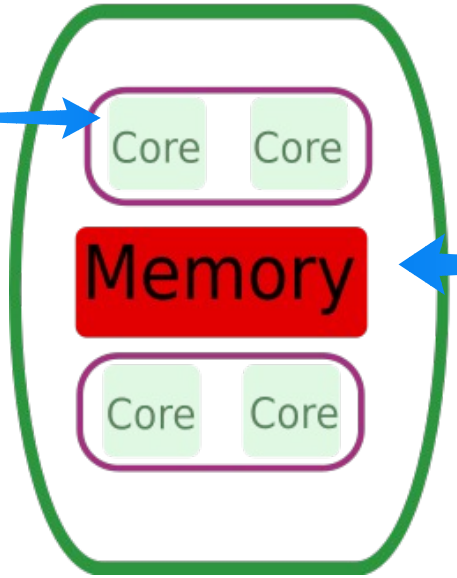


Domain Decomposition



The Compute Node

4 Cores for computation



Memory to store data

Shared memory system:

any core can access the whole memory

The Compute Node

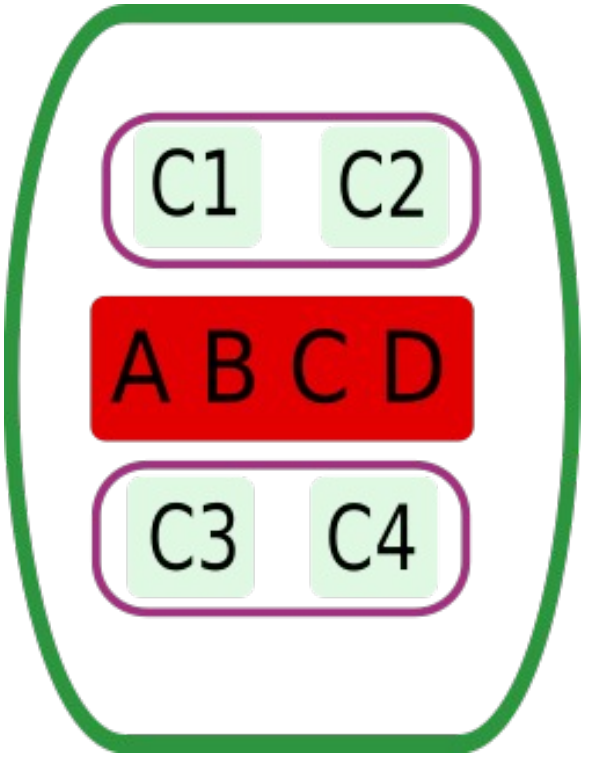
Example of a parallel set of operations

Core 1: A = A + 1

Core 2: B = B * 2

Core 3: C = C / 3

Core 4: D = D * D

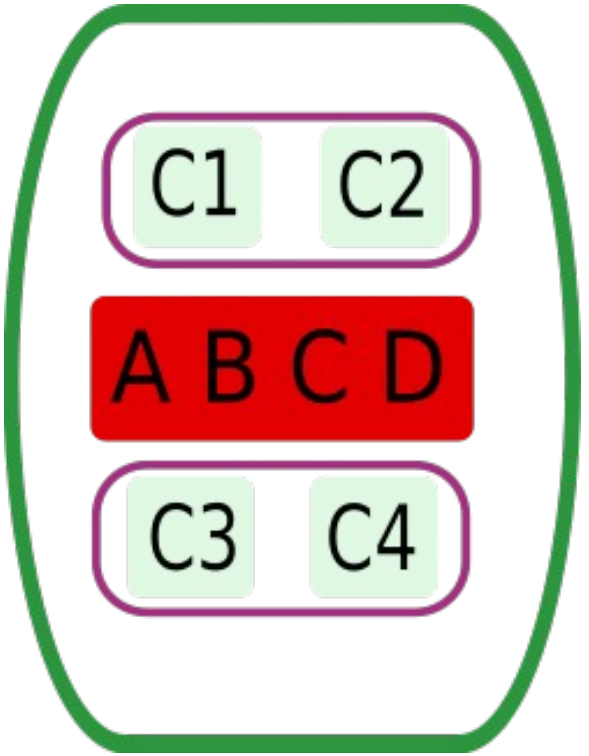


Compute node

The Compute Node

Example of a parallel set of operations

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

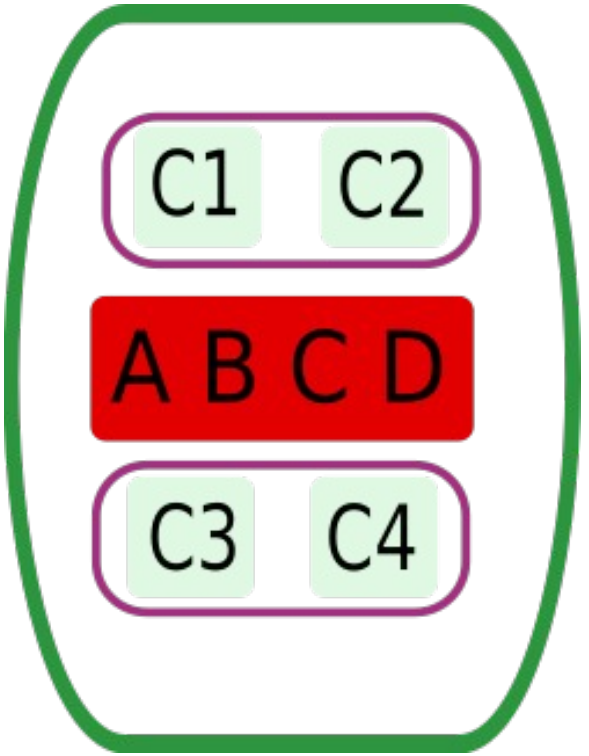


Total time of operation = 15 s

The Compute Node

Example of a parallel set of operations

1) A = A + 1
2) B = B * A
3) C = C / A
4) D = D * A

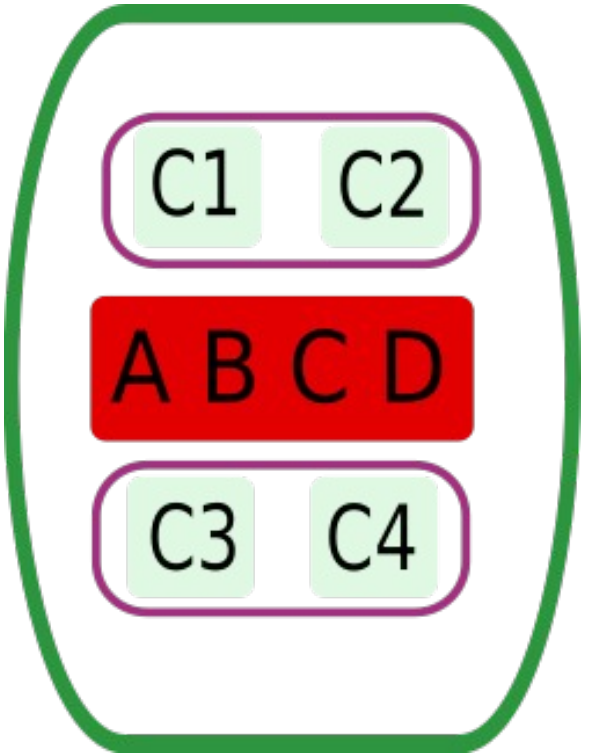


Data dependency warning:

Results depends on the order of the operations !!

The Compute Node

Example of a parallel set of operations



1) A = A + 1 (10 sec)

(Wait)

2) B = B * A (10 sec)

3) C = C / A (15 sec)

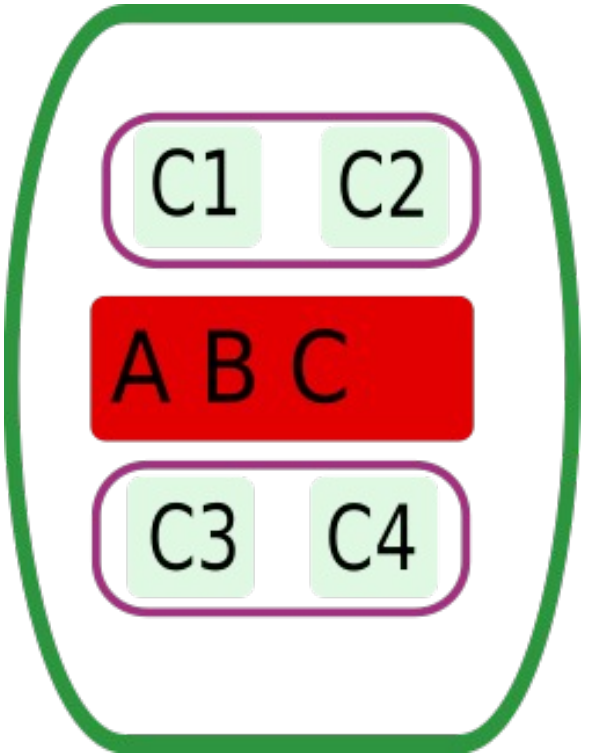
4) D = D * A (10 sec)

Total time of operation = **25 s** => dependencies are expensive because they force synchronizations.

The Compute Node

Example of a parallel set of operations

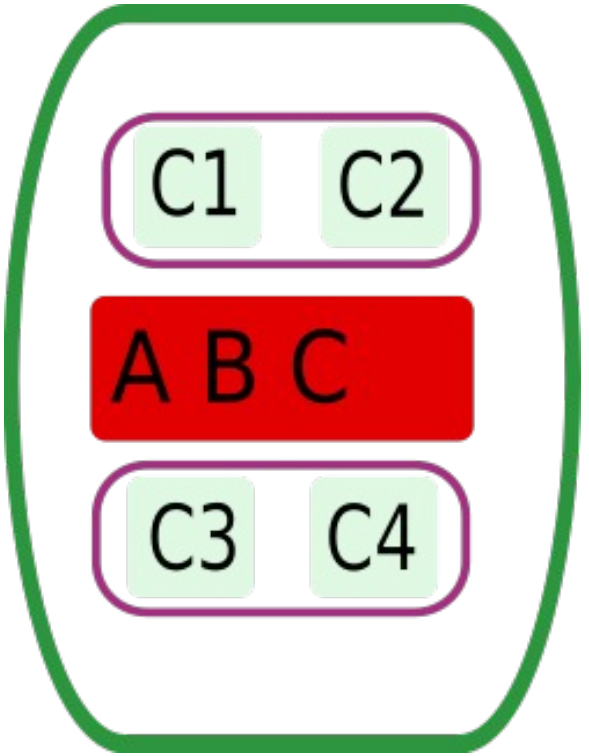
1) A = A + 1
2) B = B * 2
3) C = C / 3
4) None



The Compute Node

Example of a parallel set of operations

1) A = A + 1 (10 s)
2) B = B * 2 (10 s)
3) C = C / 3 (15 s)
4) None



Total time of operation = 15 s

Under loaded memory leads to idle cores

The Compute Node

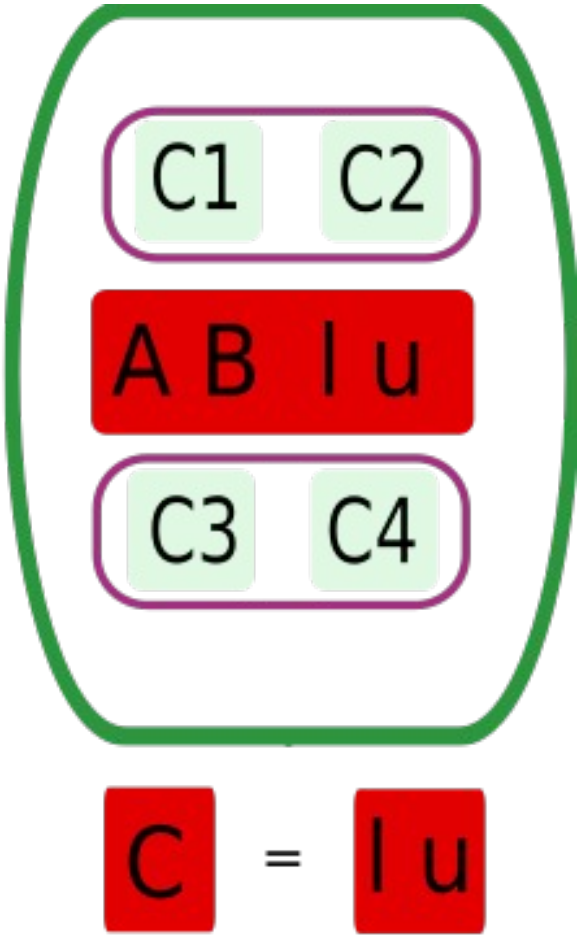
Example of a parallel set of operations

Split C into l & u

1) A = A + 1 (10 s)
2) B = B * 2 (10 s)
3) l = l / 3 (15/2 = 7.5 s)
4) u = u / 3 (15/2 = 7.5 s)

Total time of operation = 10 s

Exposing parallelism helps



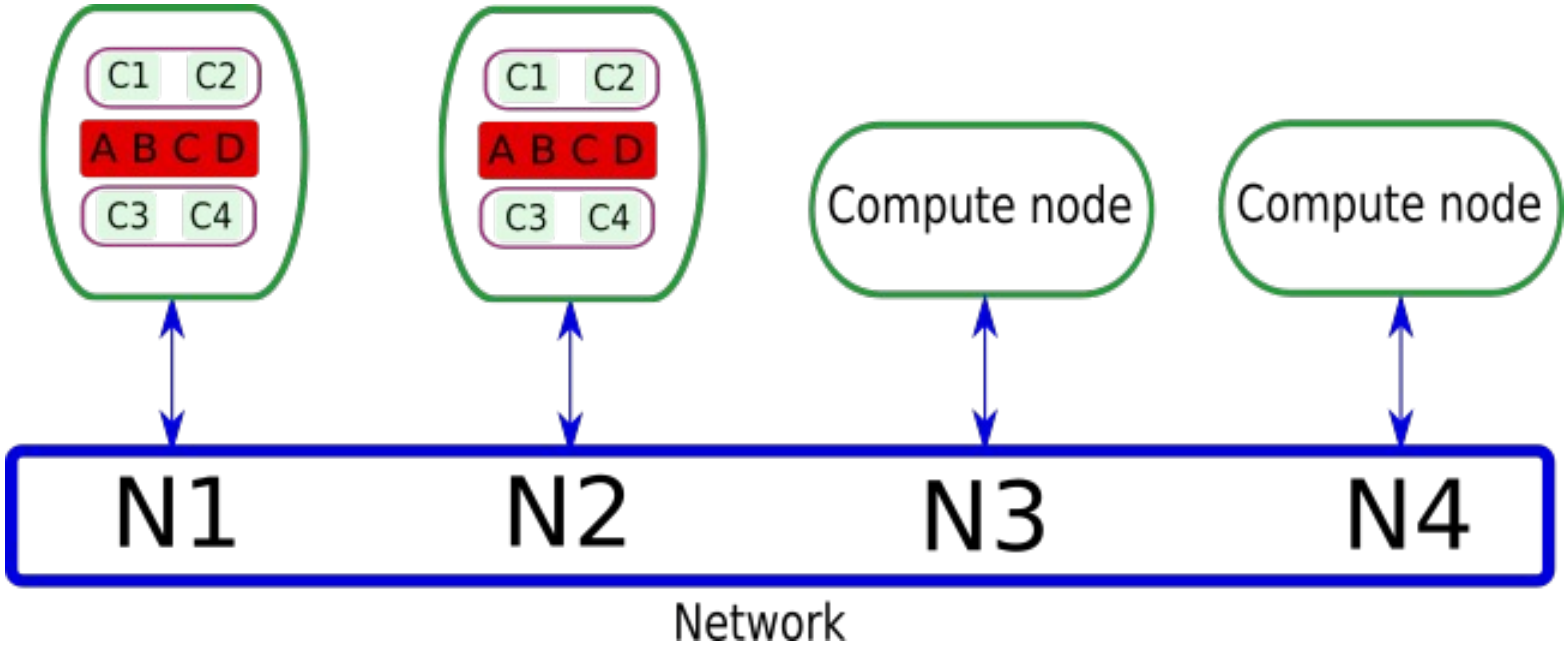
Basic rules

1) Operating on **independent data** leads to better performances because it **avoids synchronization**.

2)You need enough data in memory in order to “feed” all cores in order to **avoid idle units**.

3) You need to **expose parallelism** by separating independent data explicitly in memory.

Supercomputer



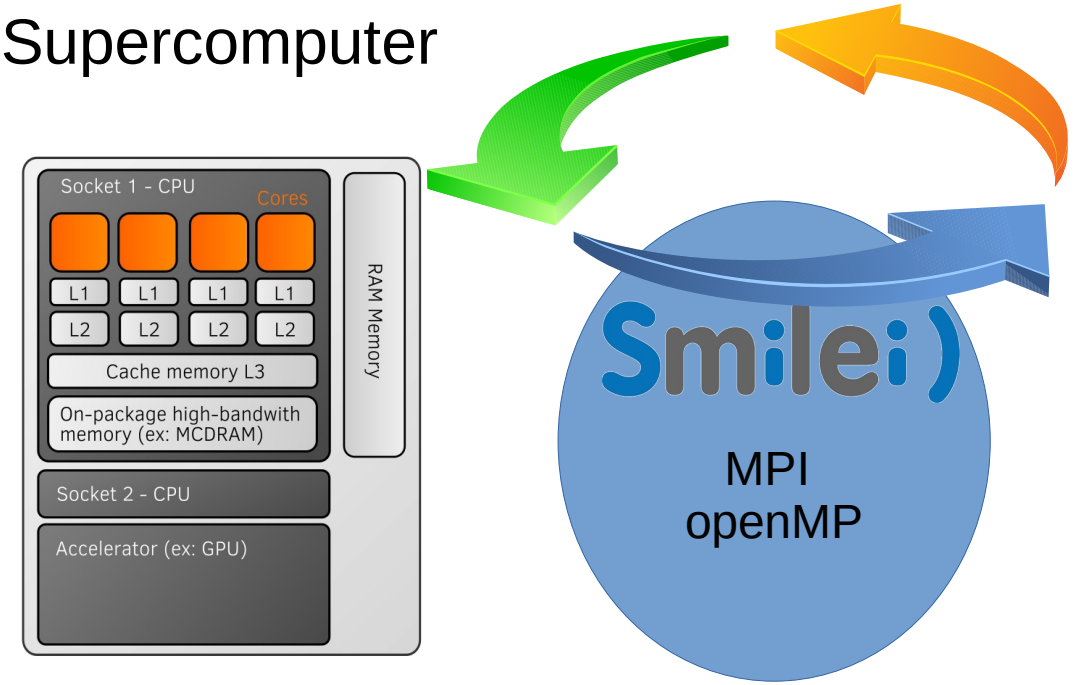
Distributed memory:

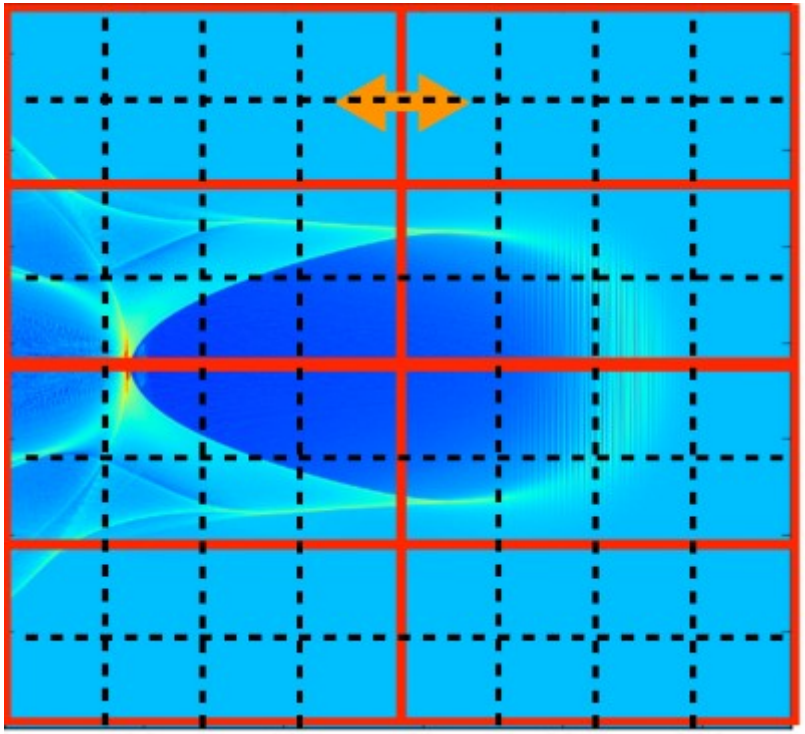
- Why ? More cores and more memory.
- But … Synchronizations need to go through the network.
- You will have to chose a number of nodes for your simulation.

Outline

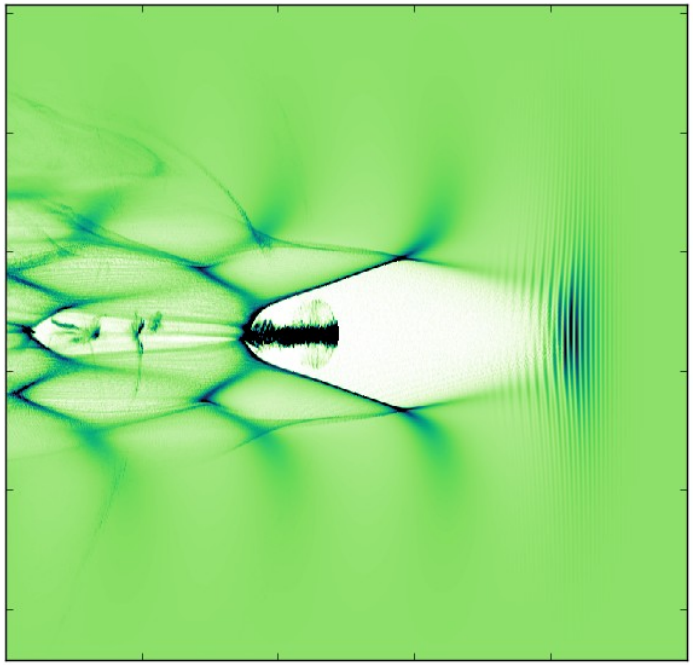
Domain Decomposition

Supercomputer

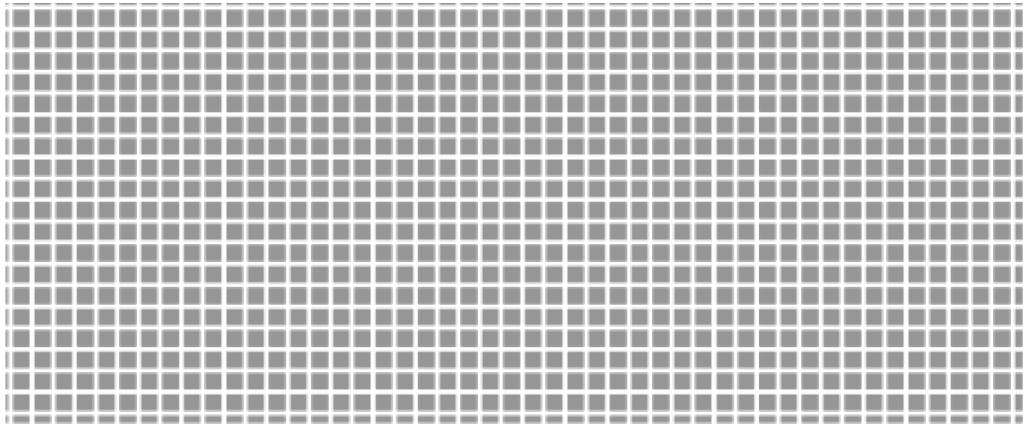




Domain Decomposition: Cells



Visualization of the data



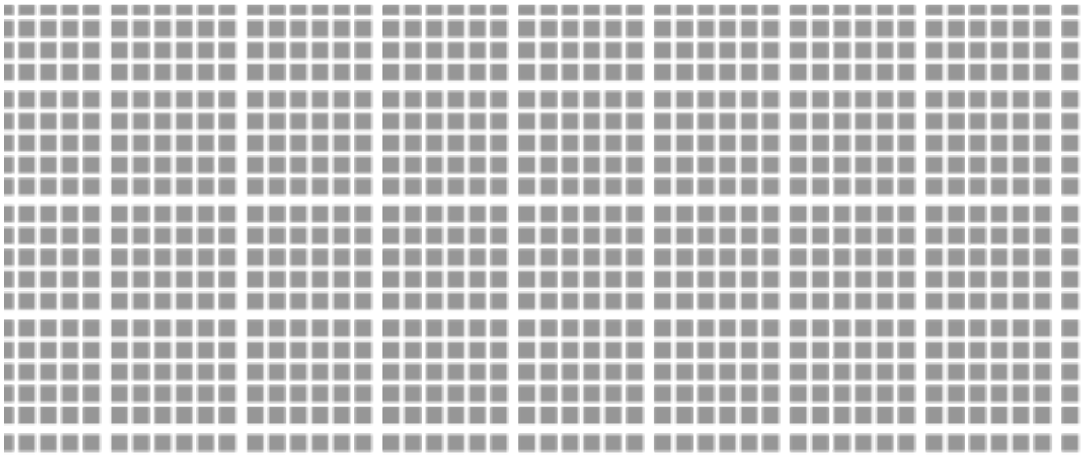
Actual data grid made of **Cells:**

- Constant number of fields
- Variable number of particles

Domain Decomposition: Patches

Let’s expose parallelism by building data sets as independent as possible.

The physics says a cell depends on its neighbors so it makes sense to group them spatially.



Patches are groups of **Cells** contiguous in space

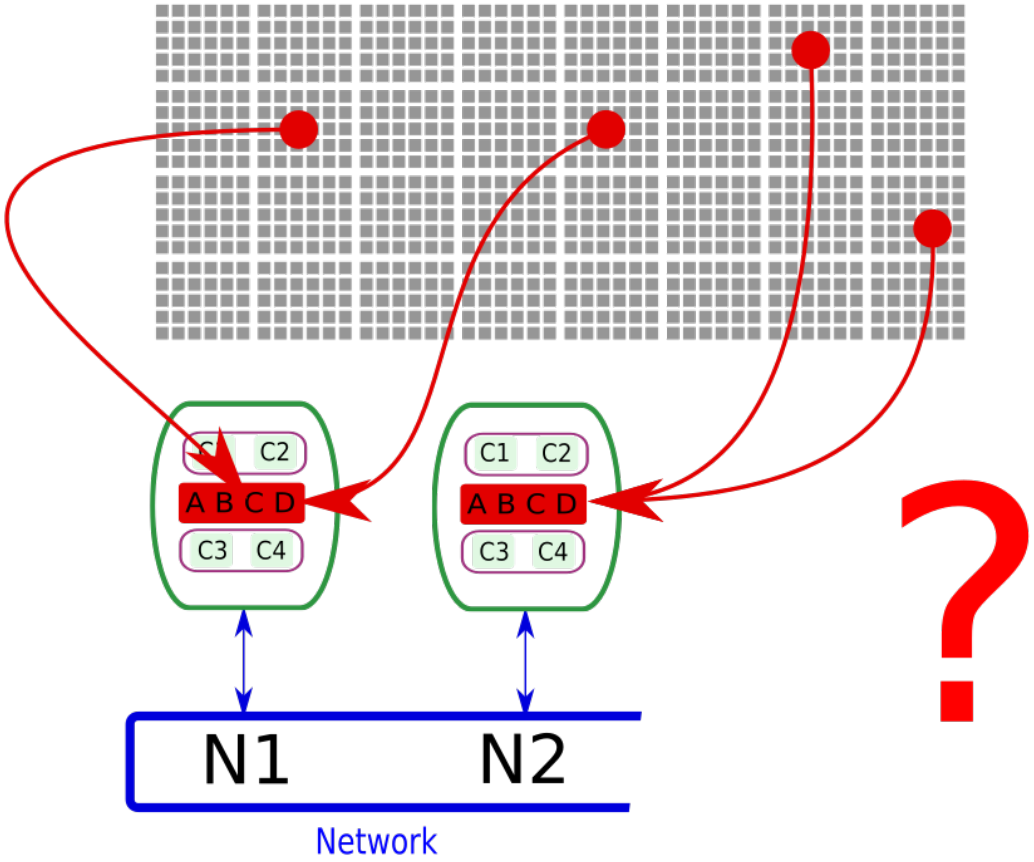
Domain Decomposition: Patches

Remarks on patches:

- Domain Decomposition => The simulation domain is divided into sub-domains called “Patches” in Smilei.
- Patches include all fields and macro-particles of their cells.
- Patches are not completely independent. Synchronization are necessary at patches boundaries.

Domain Decomposition

We now need to assign the patches to the different compute nodes.



Domain Decomposition

Guidelines to assign the patches to the different compute nodes.

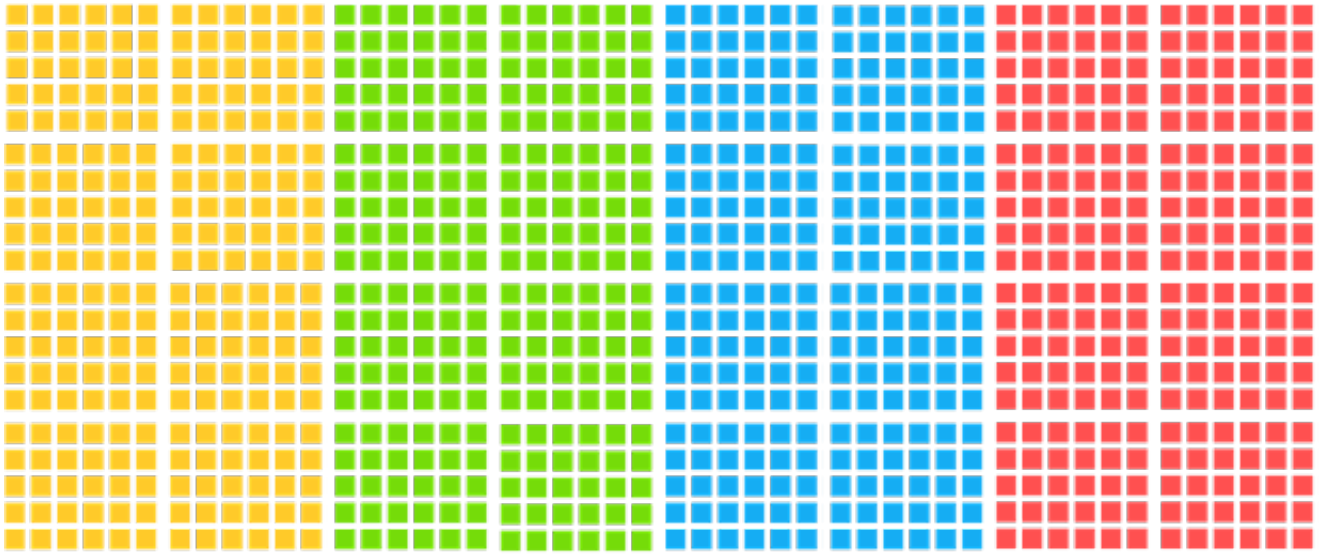
1) Patches on the same node should be **contiguous** in order to minimize synchronization via the network.

2) There should be roughly the same number of particles on each node to **balance the computing load**.

3) Some degree of **adaptability** in the patch distribution.

Domain Decomposition

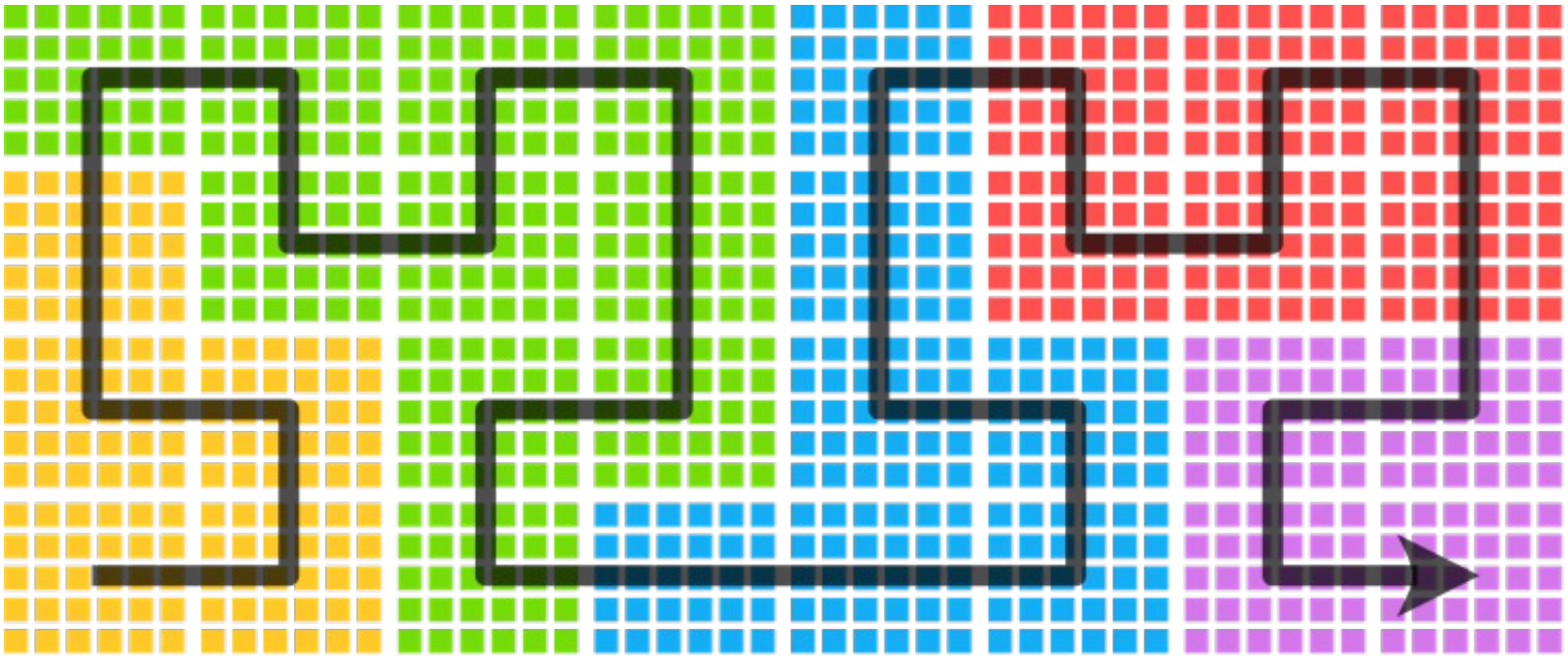
Cartesian decomposition



Possible in Smilei but lack of adaptability (not default behaviour)

Domain Decomposition

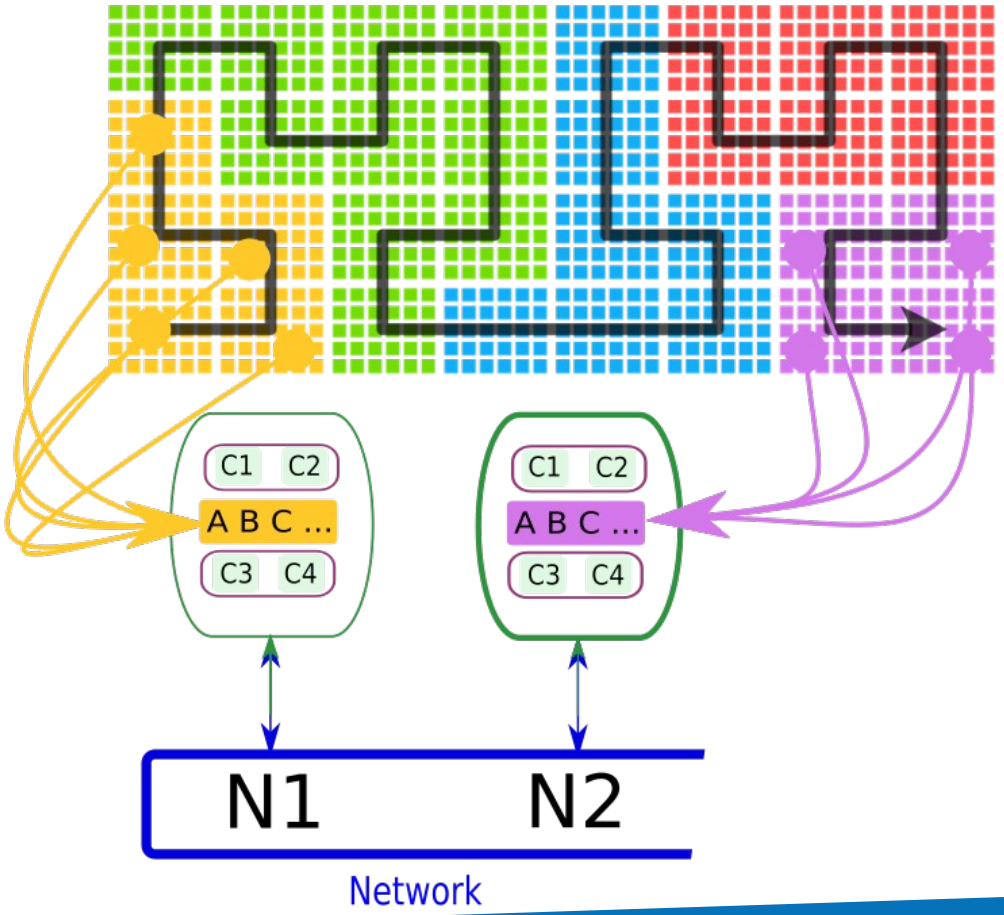
Hilbertian decomposition: Successive pieces of a Hilbert curve



This procedure guarantees to have contiguous patches.

Domain Decomposition: MPI Domain

Patches assigned to a common data set form a **MPI Domain**.



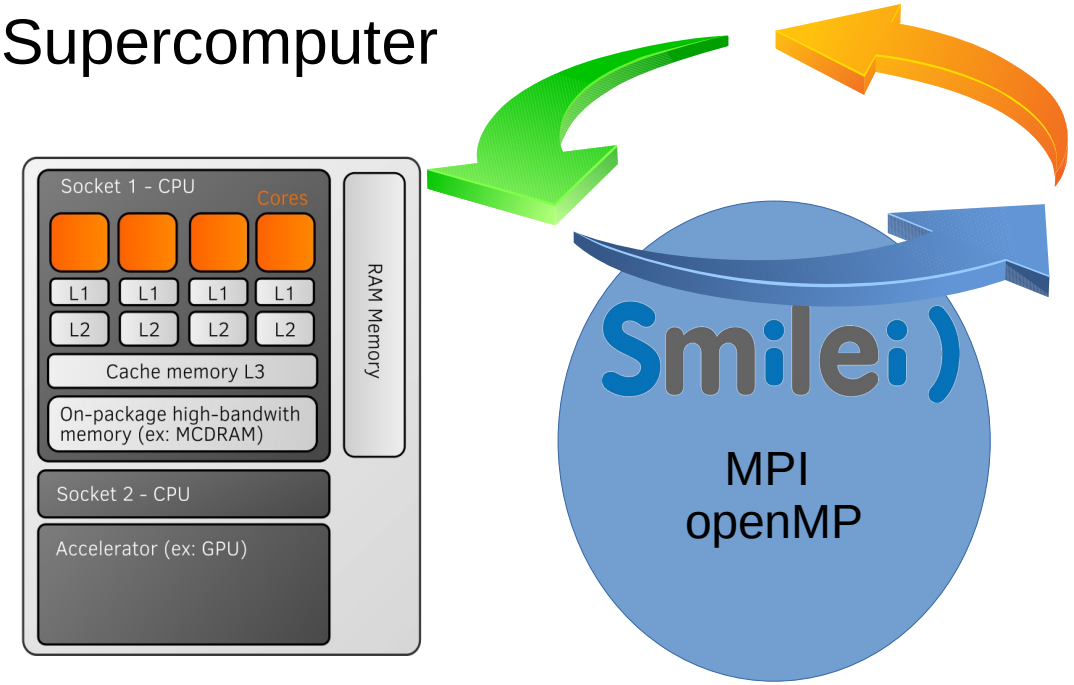
Domain Decomposition: MPI Domain

Remarks on MPI Domains:

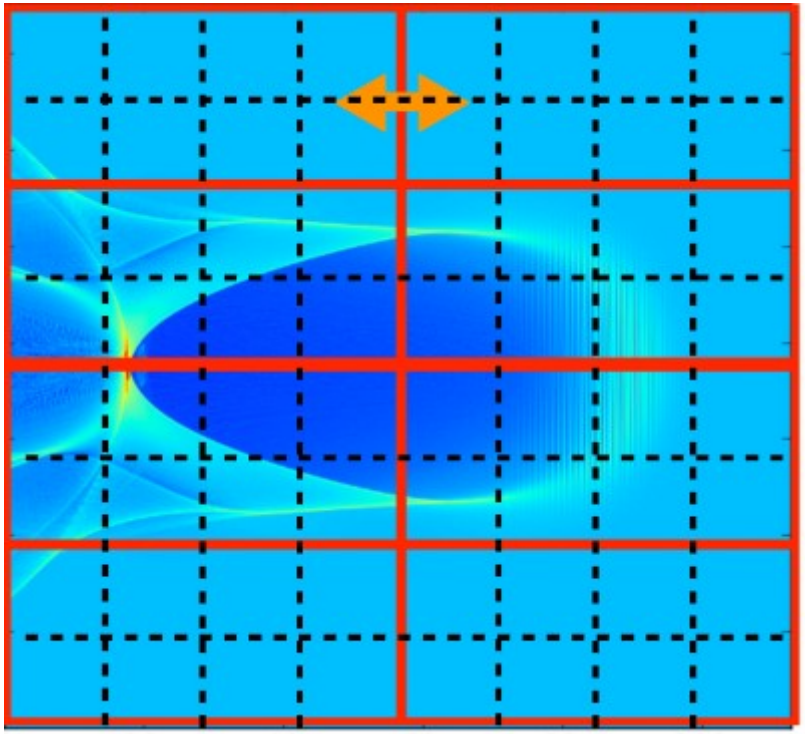
- MPI is the protocol used to handle communications on the network.
- Several MPI domains can be located on a single compute node.
- The number of patches per MPI domain is variable.

Outline

Supercomputer

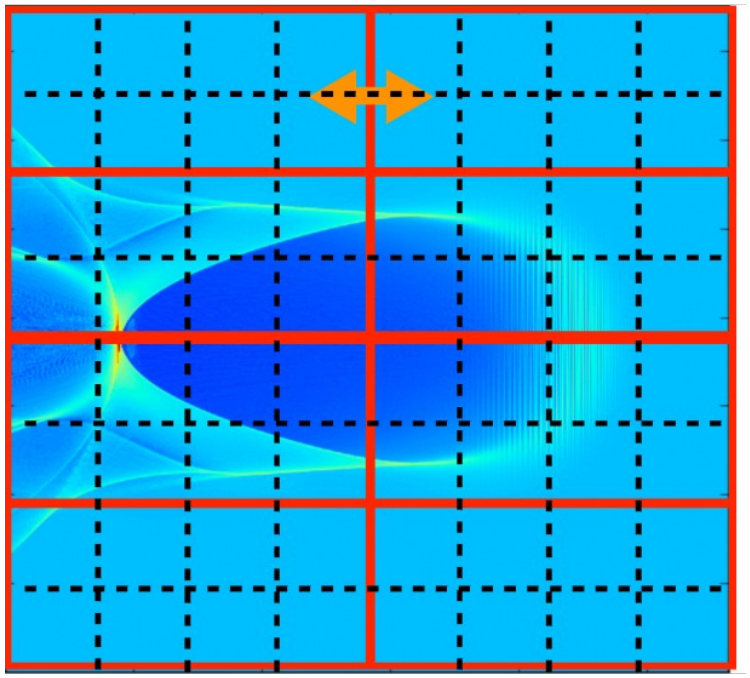


Domain Decomposition



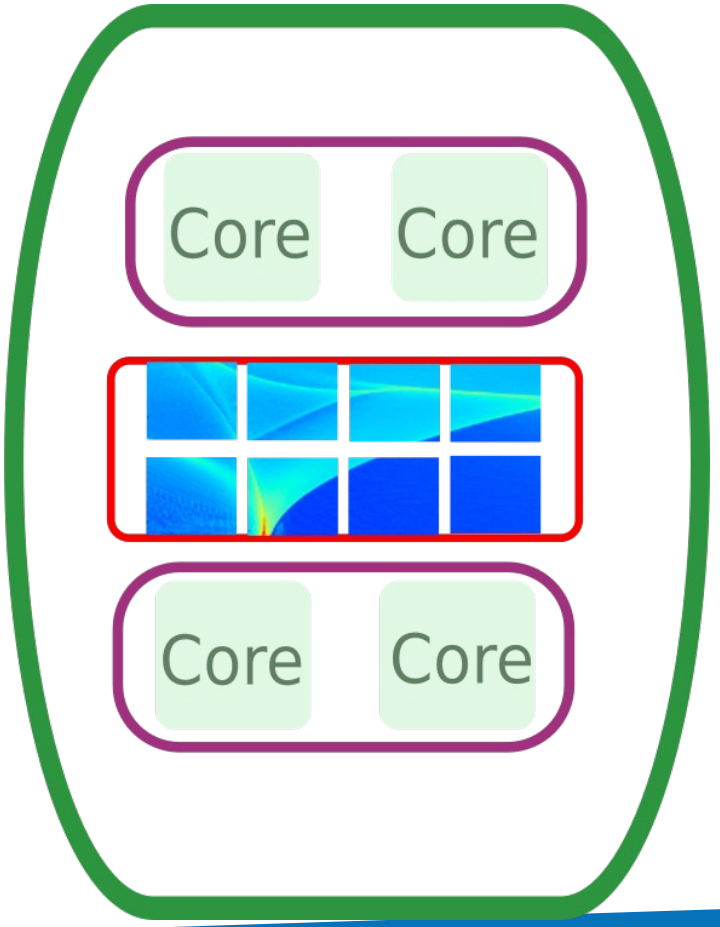
MPI + openMP setup

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



Patches boundaries

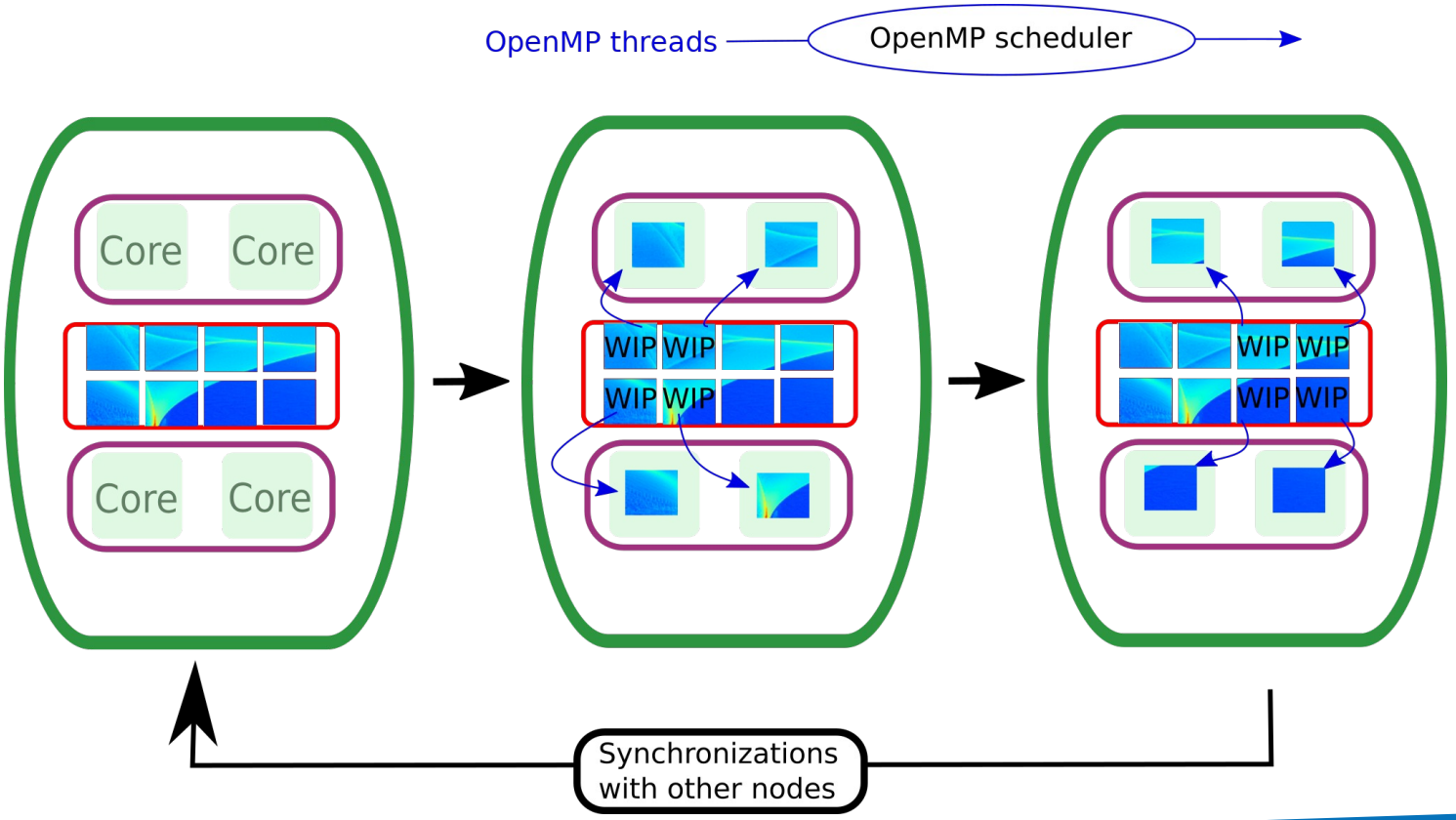
MPI domains boundaries



OpenMP threads

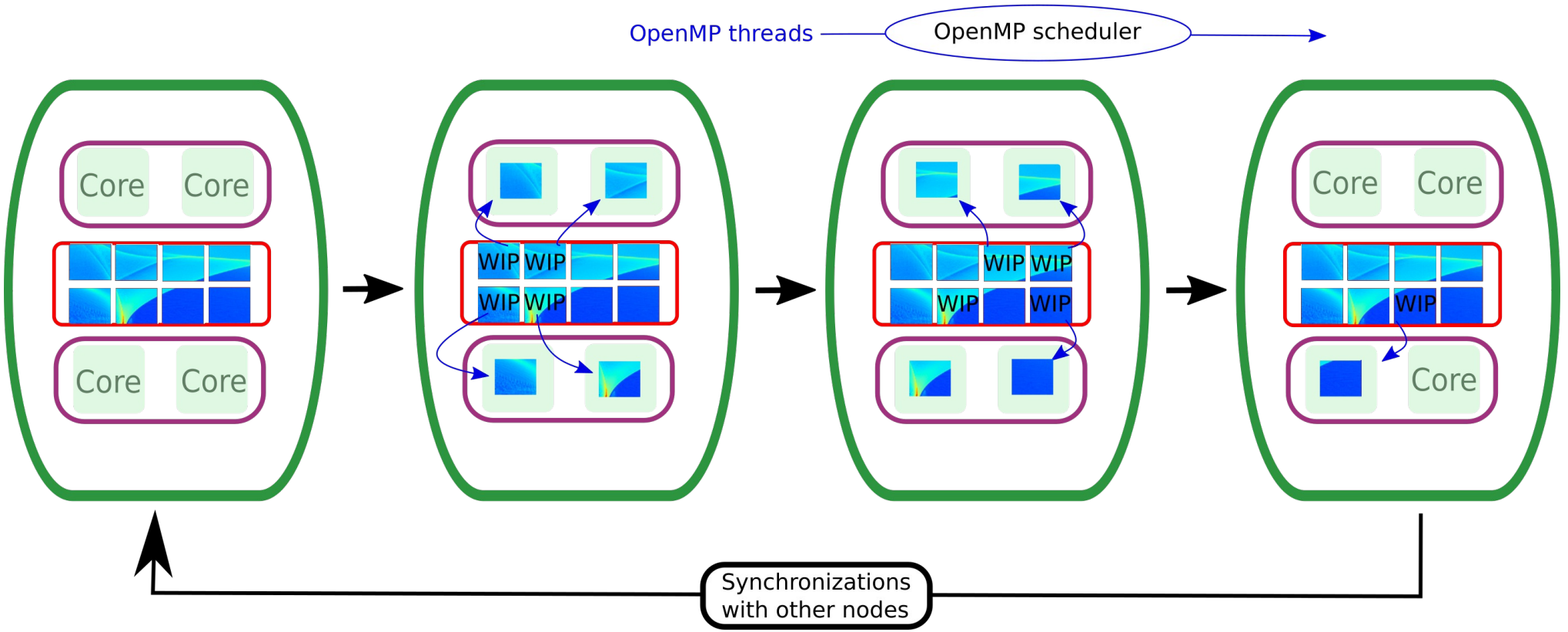
The OpenMP scheduler assigns cores to patches via the **openMP threads**.

The number of openMP threads is fixed by the user and should be **one per core**.



OpenMP threads

Heterogeneity of patch loads induces idle time.



Load Balancing on Node

OpenMP dynamic scheduler balances the load

As soon as a core becomes available, the scheduler “feeds” him with a patch => no idle time.

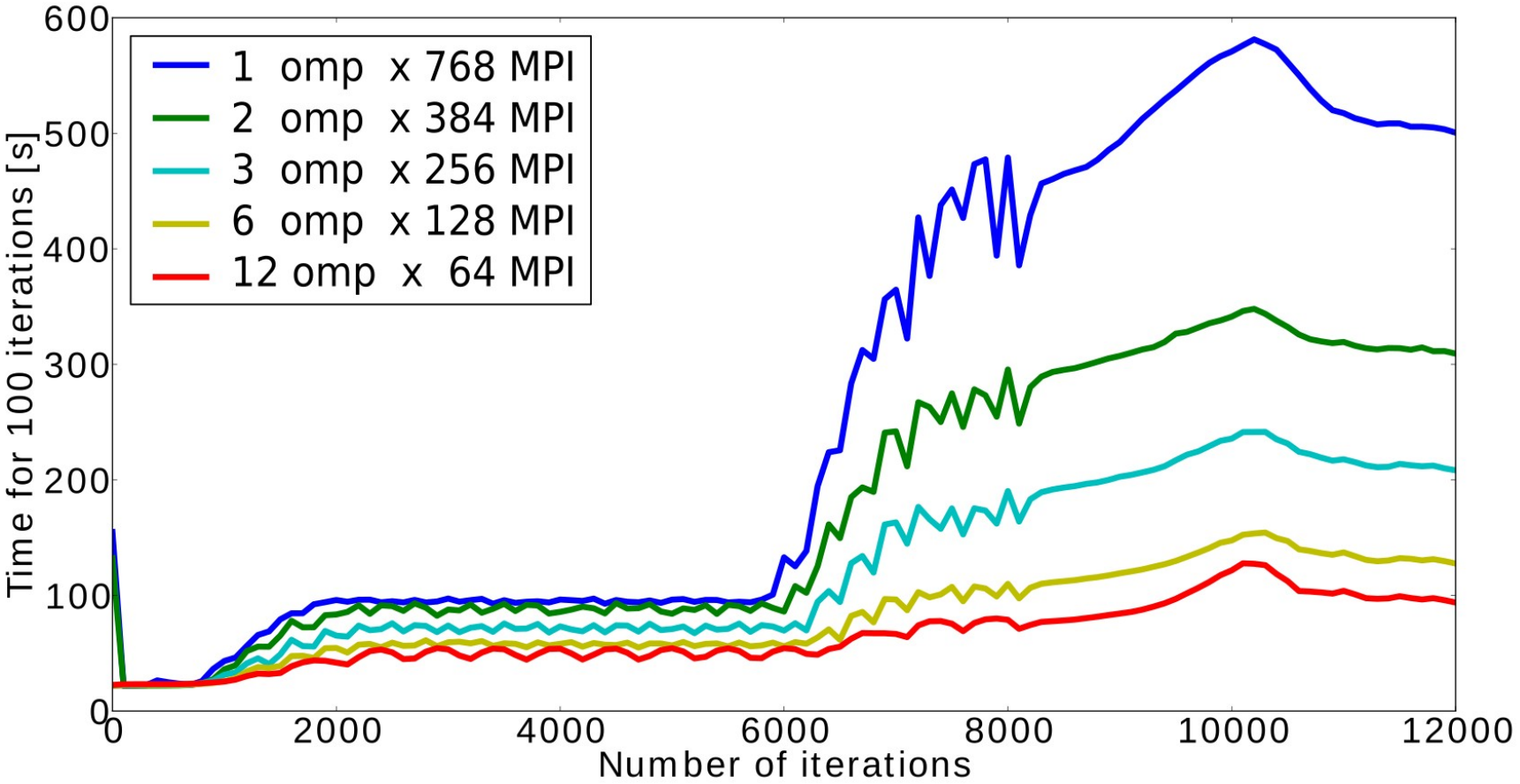
More threads, better balance.

Number of threads limited by number of cores (no hyper threading).

Works only if many more patches than threads in order to hopefully average the load.

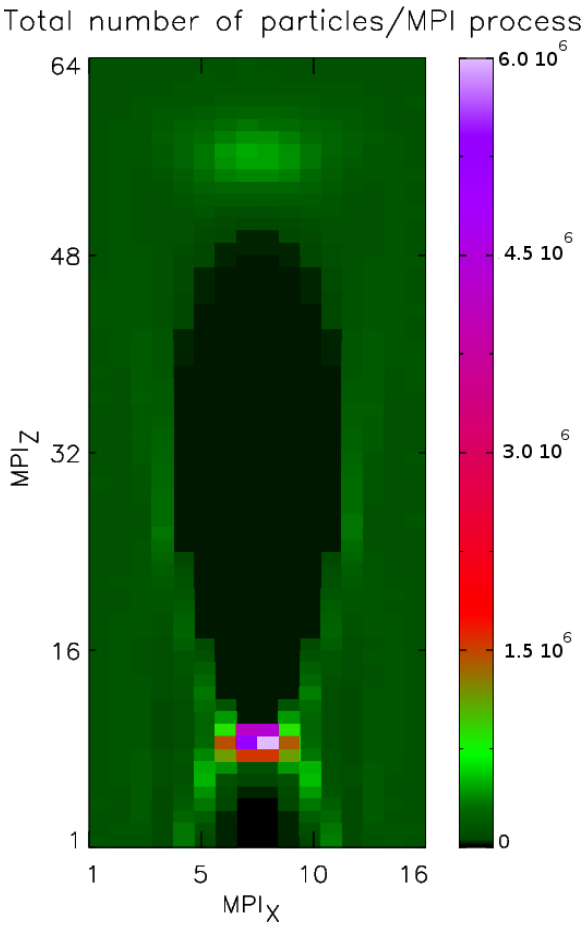
Load Balancing on Node

OpenMP dynamic scheduler smooths the load



Dynamic Load Balancing between MPI

Imbalance also occurs between compute nodes

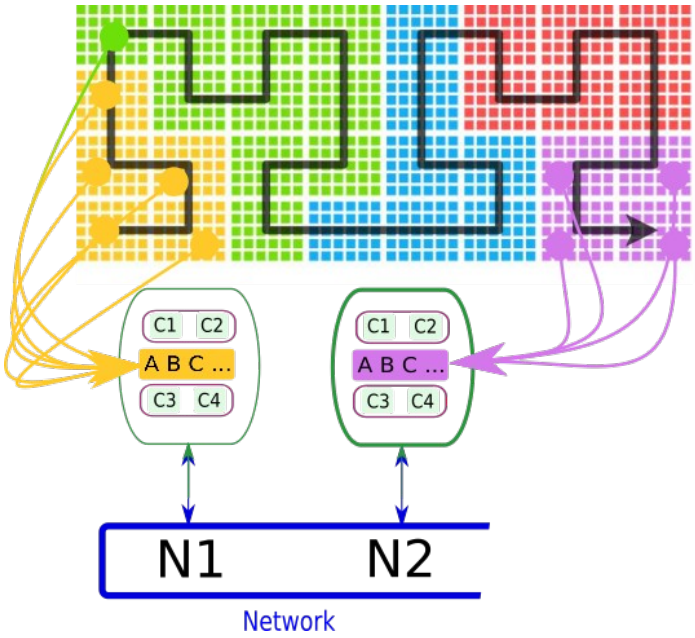


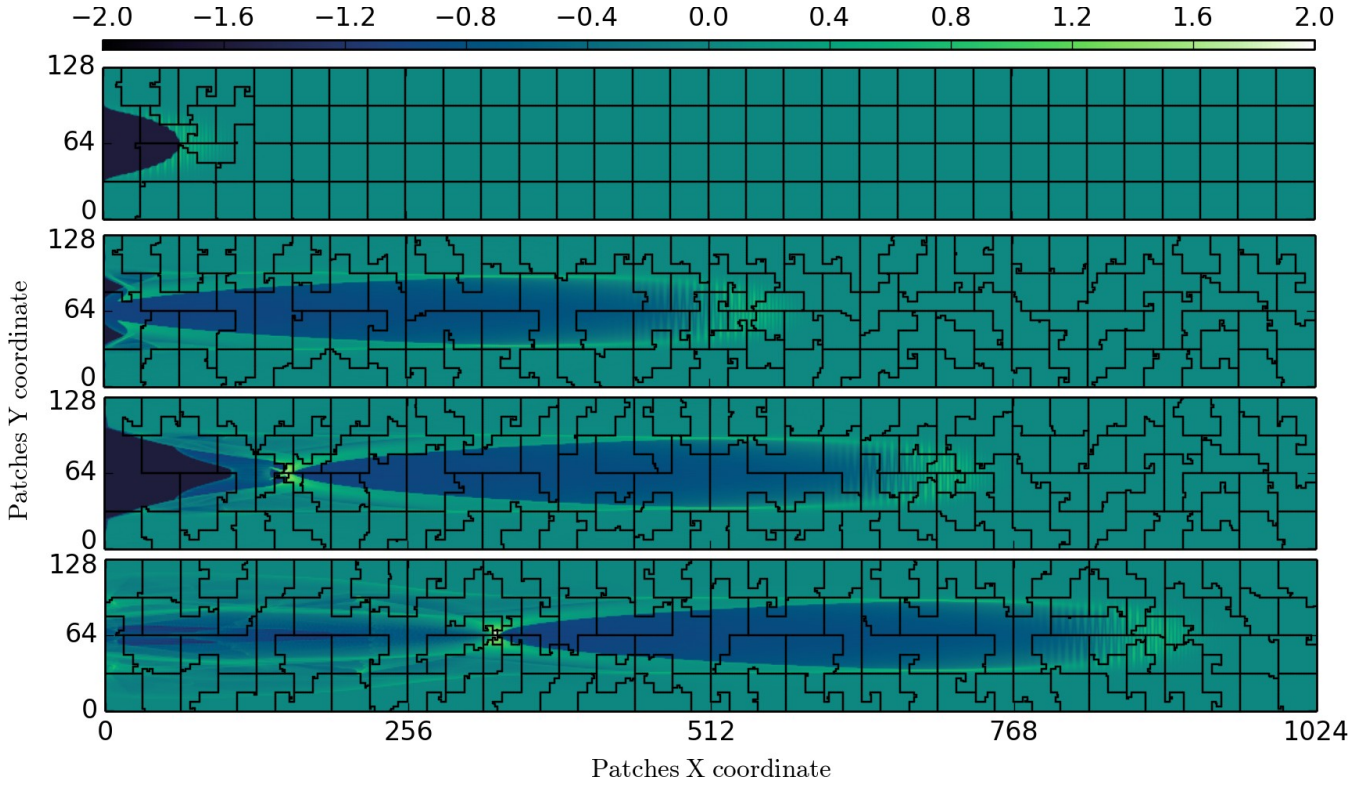
At first order, computing load is defined by the total number of particles.

Particles are not bounded and this number can evolve in time.

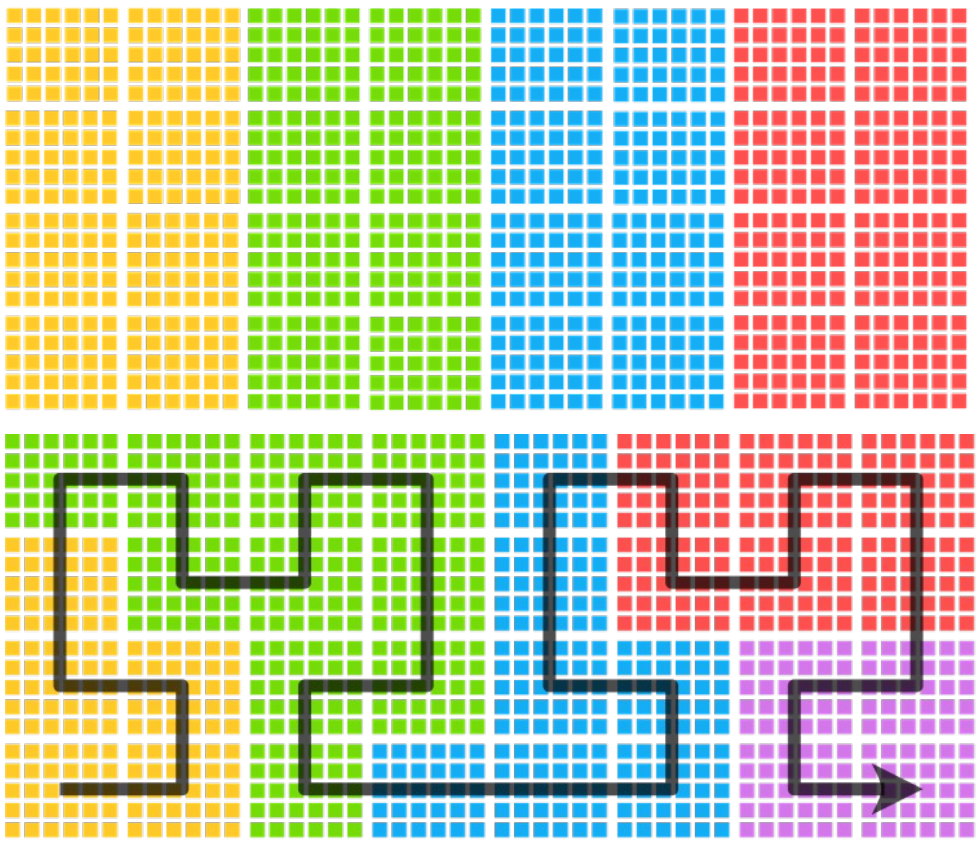
Dynamic Load Balancing between MPI

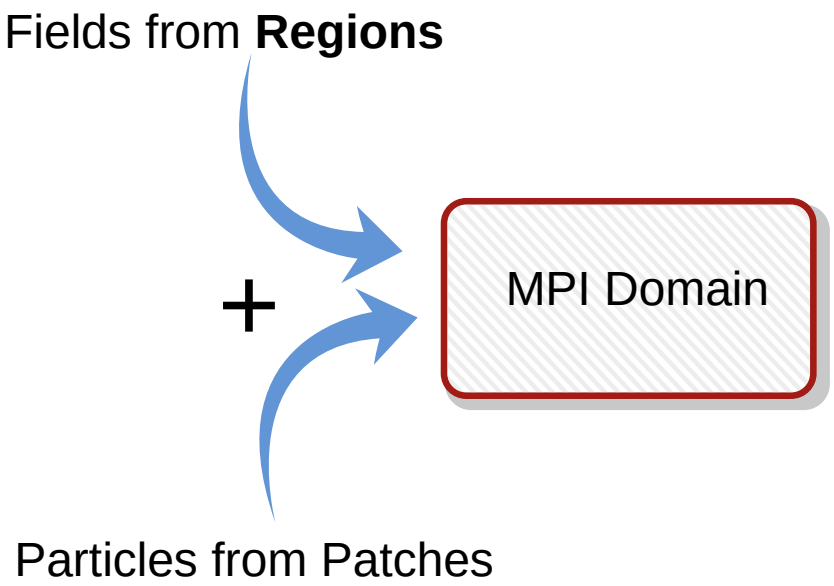
Patches are exchanged between MPI domains





SDMD: Single Domain, Multiple Decomposition





Use small patches without paying heavy synchronization costs !
Efficient operations on fields like multi-pass current filters.

Glossary

Hardware	Software	Domain Decomposition associated data
Node	MPI process	Group of patches + 1 Region (if sdmd)
Core	OpenMP thread	1 Patch

F.A.Q. about parallelization

Which MPI + openMP setup should I use ?

- 1 MPI per socket (usually 2 per node)
- 1 openMP thread per core on socket
  export OMP_NUM_THREADS = ...
- Use openMP dynamic scheduler
  export OMP_SCHEDULE = dynamic

F.A.Q. about parallelization

How many patches should I use ? OR What size should my patches be ?

1) **At least 1 patch per openMP thread.**
   Warning: this rule applies for each MPI process and at all times.

2) **There is a minimal size of patch depending on interpolation order.**

3) In general you need: N patches > 4 N omp.

4) **Do your tests !** More patches helps balancing the load but induces synchronization overhead.

Perspectives

- Asynchronism: overlap of communication and computation.
- Incoming GPU: a different memory architecture.
- Task based implementation: expose finer grain parallelism (see Mathieu’s talk).
- Exascale => importance of node level optimization (see Mathieu’s talk).

Acknowledgements

Thank you for your attention!

Funding agencies

PLAS@PAR Plasmas à Paris

P2IO Physique des 2 Infinis et des Origines

PALM Laboratoire d'Excellence Physique : Atomes Lumière Matière

Physique des Infinis Une Initiative Sorbonne Université

AIDAS AI Data Analytics and Scalable Simulations

Contributing labs & institutions

MAISON DE LA SIMULATION

LLR

LPP

irap astrophysique & planétologie

cnrs

ÉCOLE POLYTECHNIQUE

INSTITUT POLYTECHNIQUE DE PARIS

IDRIS

intel

LSI

CELIA

cea

SORBONNE UNIVERSITÉ

université PARIS-SACLAY