An industrial tool for scientific computing: PABLO - PArallel Balanced Linear Octree

Marco Cisternino, Institut de Mathematiques de Bordeaux, OPTIMAD Engineering Srl
Edoardo Lombardi, OPTIMAD Engineering Srl

Introduction

PABLO is C++/MPI library for managing **parallel linear octree/quadtree**. It has been developed at OPTIMAD Engineering Srl and it has been pre-released under the **GNU Lesser General Public License**.
The aim of the project is to provide users with a tool to manage parallel adaptive grid of quadrilaterals/hexahedra avoiding burden of implementing MPI instructions in their applications code.

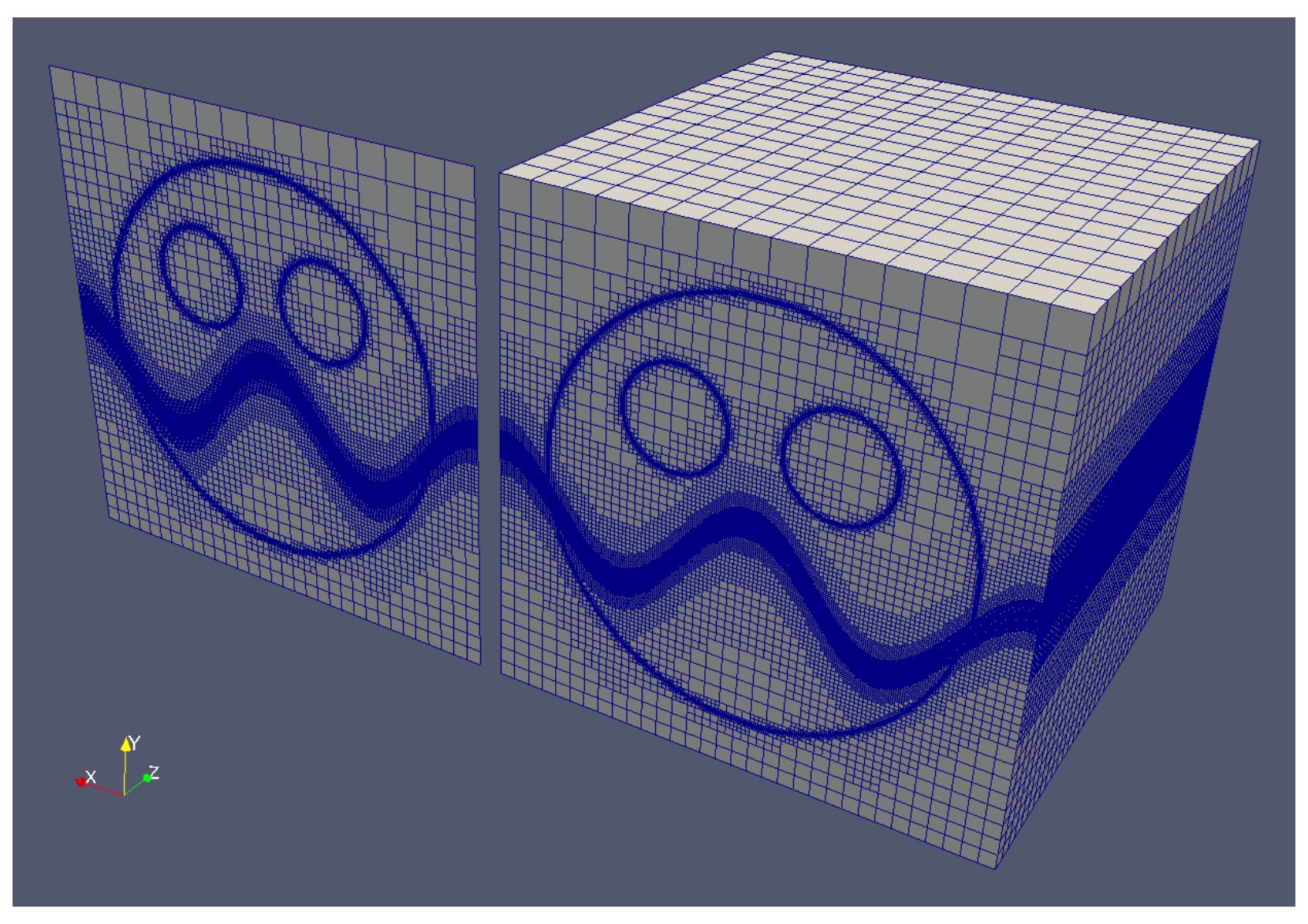

Figure 1

Template C++ classes in the dimension parameter have been used, providing the user with the opportunity of writing its application code **independently from the dimension** of the problem. Actually 2D and 3D specialization of PABLO are available.

Basic Principle I

The basic idea in PABLO is the **Morton Index**. This indexing system yields:

- a global persistent label for every element (M),
- a local non-persistent always consecutive numbering (N), called Z-curve.

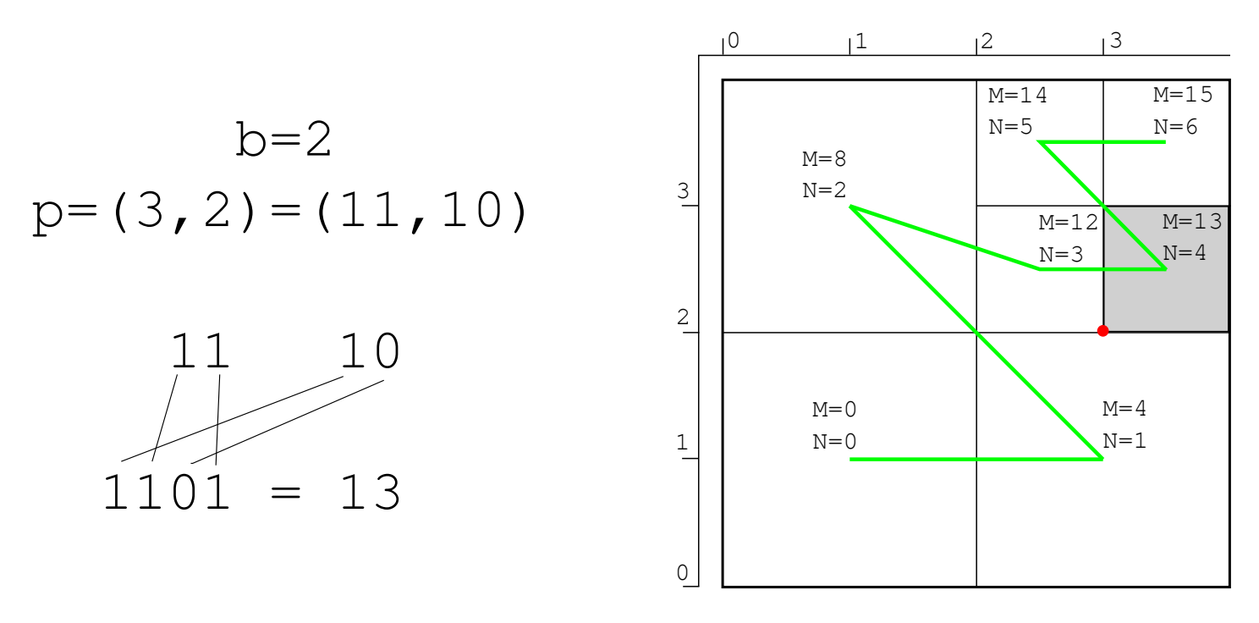

Figure 2. Interleaving example (left), Z-curve (right)

Given a max level (b) of refinement starting from an initial element of level $= 0$, a **system of global logical coordinates** for one of the element nodes can be defined. The Morton index is obtained by **interleaving** of the binary representation of these coordinates using b-bits.

Basic Principle II

The elements of PABLO are uniquely defined by their **coordinates and level**. By this way, the **memory footprint** is about ~30B per element.
Moreover, only existing element (**leaves**) are stored and a **linear container** can be used.

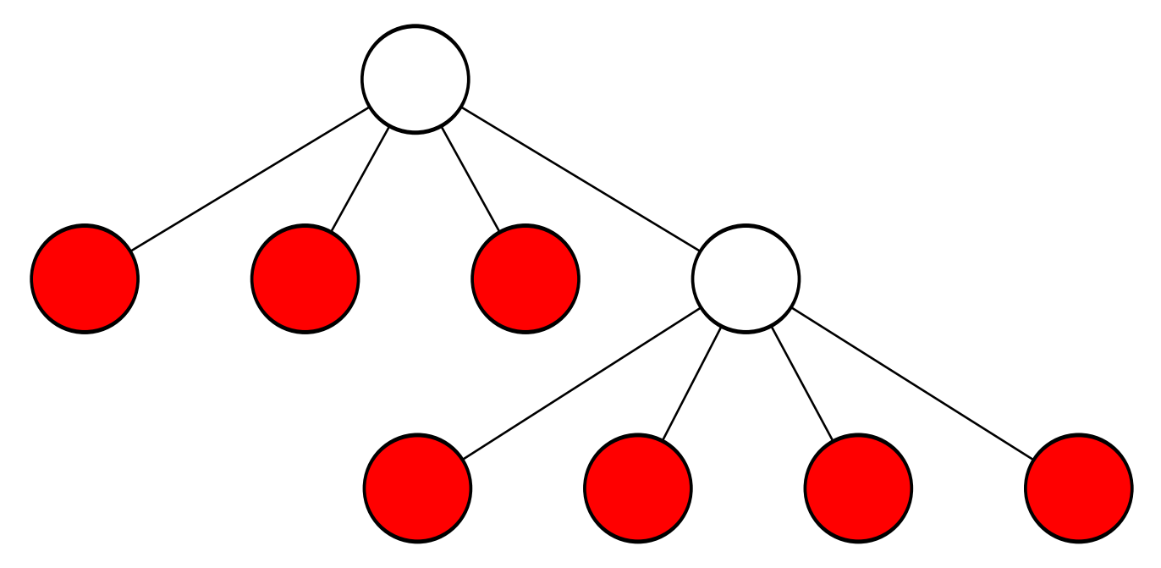

Figure 3. See Fig. 2, red is for stored elements.

The user is free to attach **any kind of data** (POD datatype or C++ available structures) to each element by using the local consecutive numbering and the favourite data container.
Finally, at the moment an embedded translation/scaling **mapping** from logical to problem space is implemented.
In future, the user should be able to introduce arbitrary mappings.

Adaptive Mesh Refinement

Both element **refining** and **coarsening** procedures are available in PABLO. Because of the possible **concurrency** and incompatibility between this two procedure, we decided to favour the refinement of elements for the sake of details resolution.
The adapting procedure is based on **markers**. Each element can be marked by a signed integer defining how many times it should refined/coarsen.

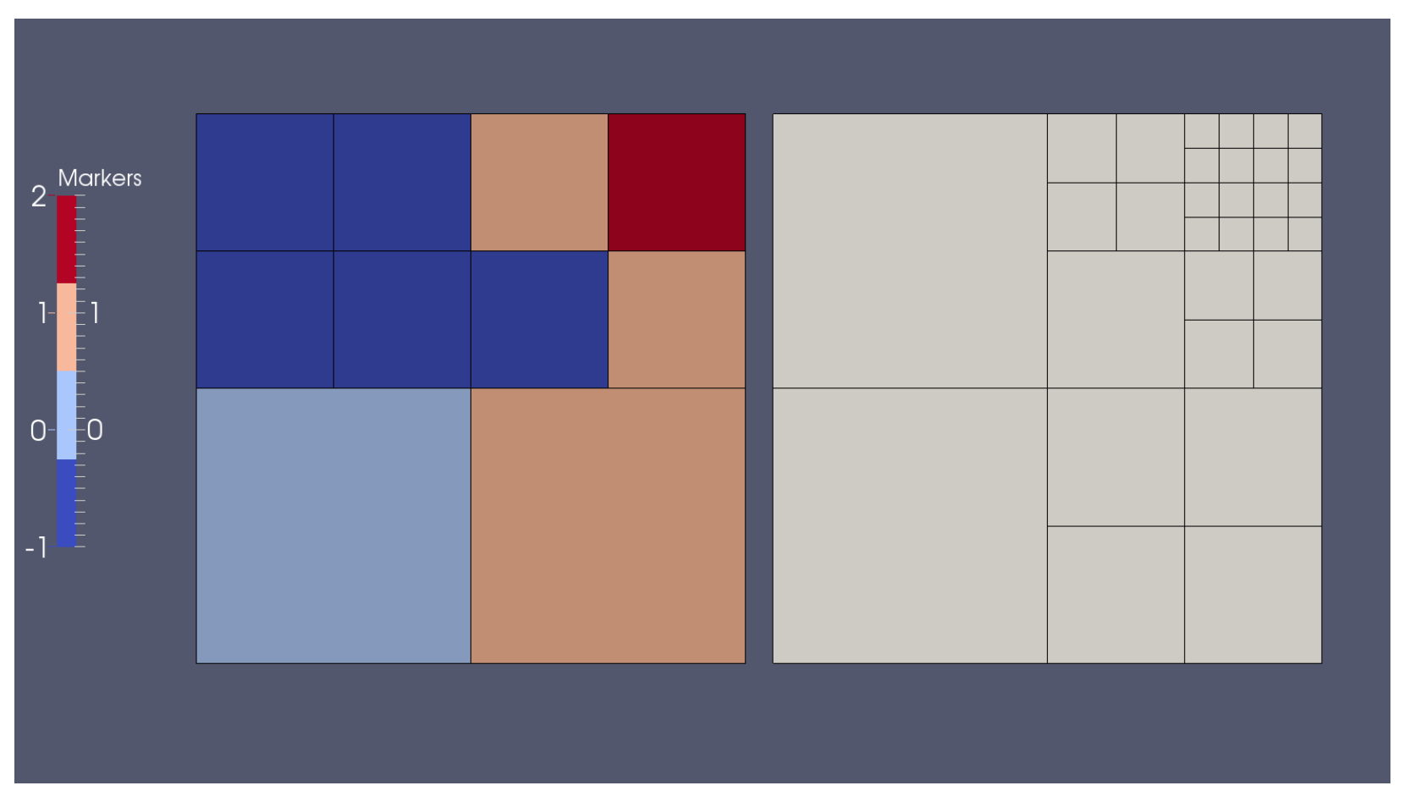

Figure 4

Therefore, a simple call to an **adapting method** changes opportunely the grid.
A **mapper** can be obtained in order to map user data from the previous to the actual grid.
Currently, maximum level of depth are 32 for 2D meshes and 21 for 3D ones.

2:1 Balance

PABLO allows the user to choose to set the **maximum level difference** between neighbours to 1 for any single element. This choice can be locally applied to the single element.
This constraint yields locally or globally **graded meshes**. The algorithm changes the user set markers using an **iterative procedure**.
The neighbours across **all codimension entities** can be balanced hierarchically, e.g. balancing across corners in 2D it would also mean balancing across faces.

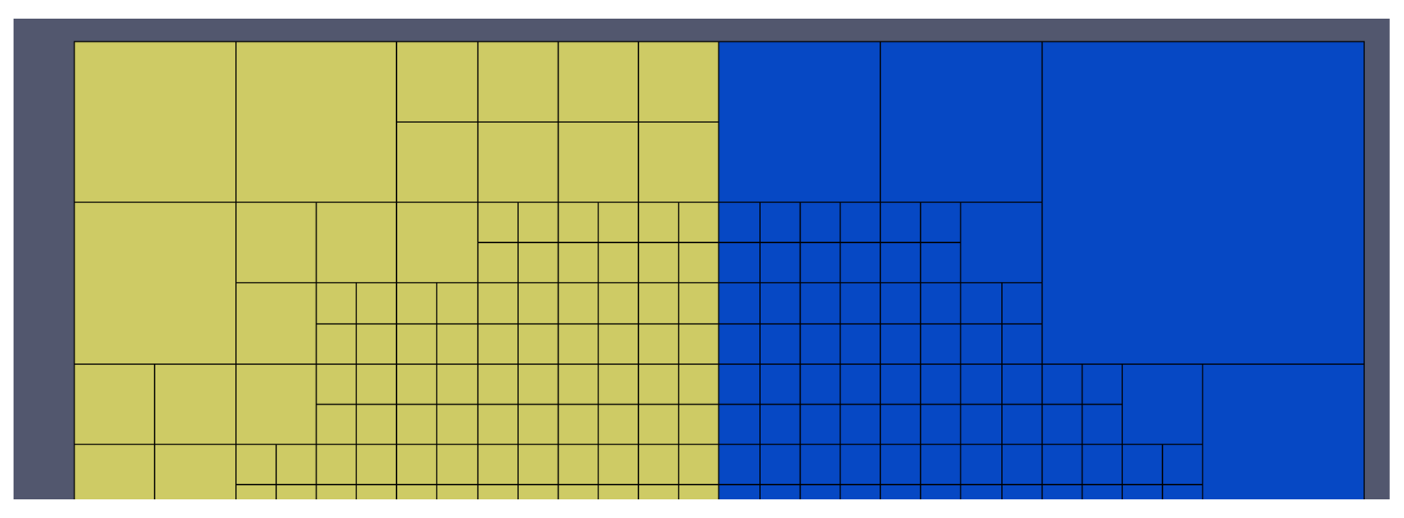

Figure 5. Balanced (yellow) - Non Balanced (blue)

Parallelism

PABLO is based on **data parallelism paradigm** using Message Passing.
The partitioning of the grid follows the ordering given by The Morton Index.
Actually, the number of elements is equally distributed among the processes.
In the future, element computational weights will be introduced in order to have computational cost partitions.

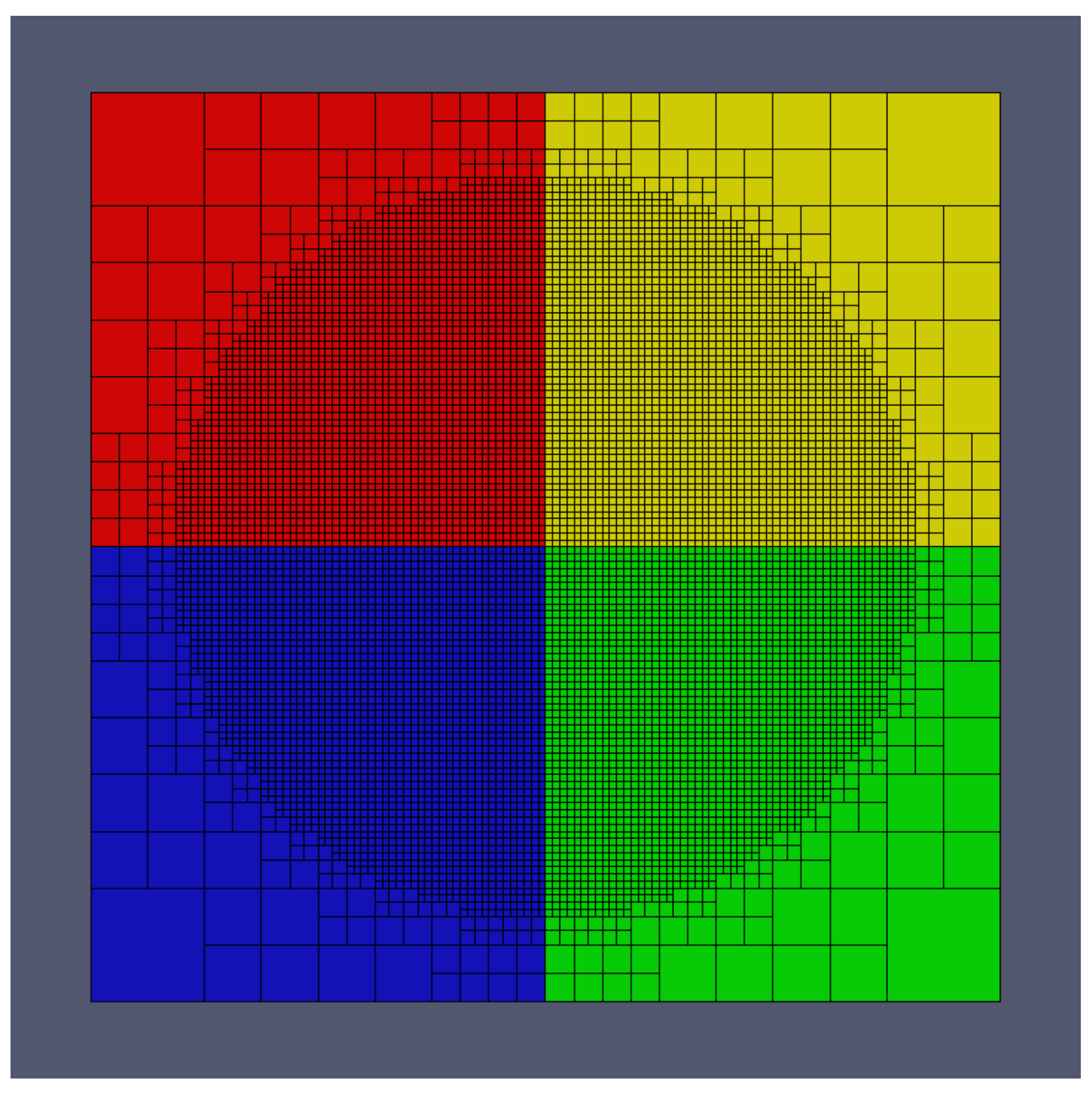

Figure 6.Different colors for different processes

The **Curiously Recurrent Template Pattern** has been used to implement **user data interfaces**. These interfaces avoid the user to explicitly call MPI routine in order to communicate data. Communications are, therefore, completely **transparent** to the user.

Dynamic Load Balance

Every time the adapting procedure yields an unbalanced partition of the elements, the use can **dynamically rebalance** the grid and the data.

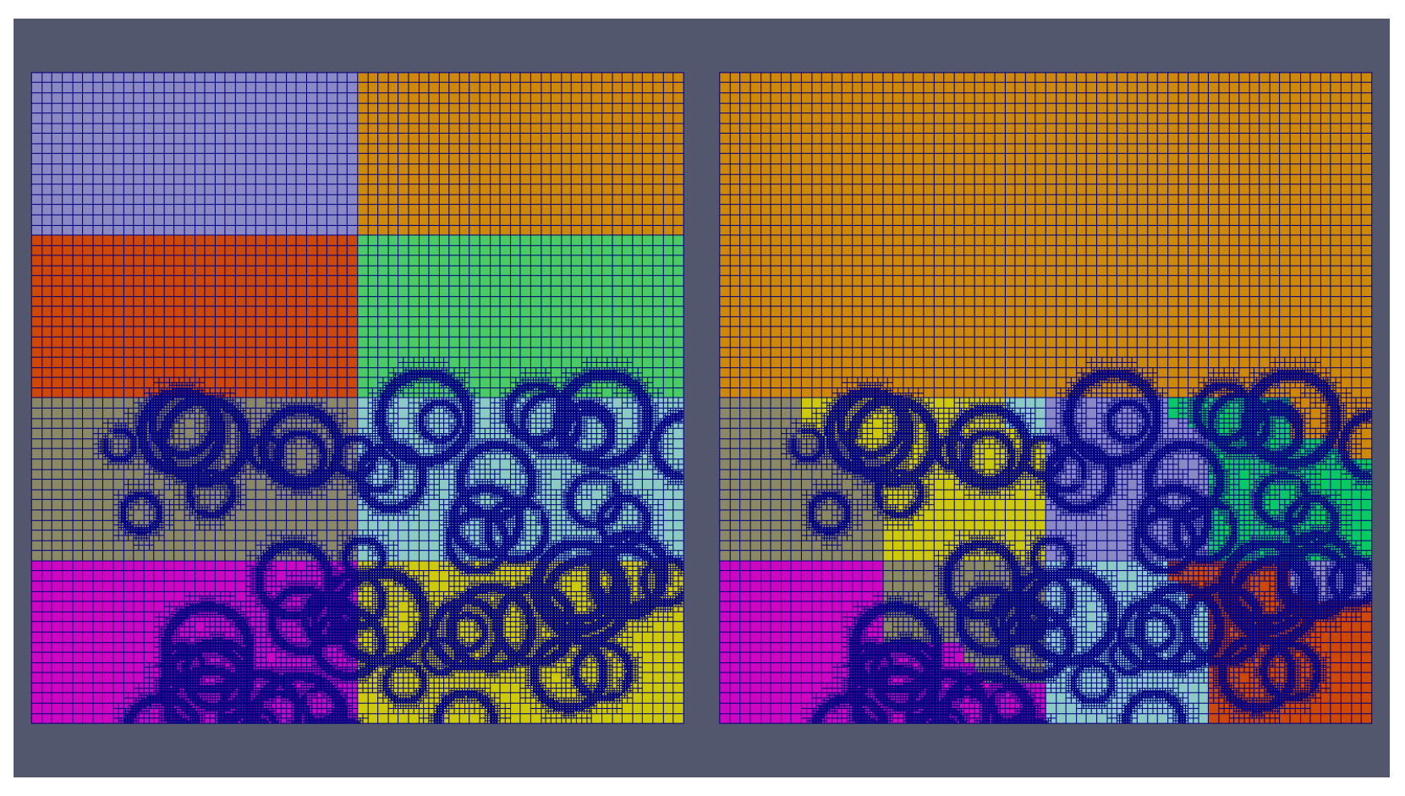

Figure 7.Different colors for different processes

Moreover, **entire families** of elements (until a user defined level starting from the finest existing one) can be kept **on the same process** in order to limit communications (e.g. projection/restriction operators in multi-level algorithms).

Current Applications in Bordeaux

Currently PABLO is being used in the following projects:

- Phase changing materials simulation (A. Raeli, M. Azaïez, A. Iollo, M. Bergmann IMB-INRIA-IPB)
- Electros, electrostrictive materials simulation (M. Cisternino, A. Colin, P. Poulin, L. Weynans, A. Iollo IMB-INRIA-CRPP)
- Python wrapping for particle flows (F. Tesser, A. Iollo, M. Bergmann IMB-INRIA)
- Polyatomic Rarefied Gas Flow (F. Bernard IMB-INRIA)

Future Developments

As soon as possible we aim to introduce **more features** and a bunch of them follows:

- periodic boundary conditions
- degrees of freedom on intersections
- arbitrary ghost layer depth
- nesting of PABLOs
- unstructured grids of PABLOs

Join the community and help us in the PABLO's development!

Contacts

Marco Cisternino, *marco.cisternino@math.u-bordeaux1.fr*
Edoardo Lombardi, *edoardo.lombardi@optimad.it*
`http://www.optimad.it/products/pablo/` -
`https://github.com/optimad/PABLO`

Documentation

PABLO's Doxygen documentation and some simple tutorials can be found here
`http://optimad.github.io/PABLO/`