Parallel remeshing and multi-grid management

4th Workshop of Project Hoscar

C. Lachat & C. Dobrzynski & F. Pellegrini
INRIA Bordeaux – Sud-Ouest

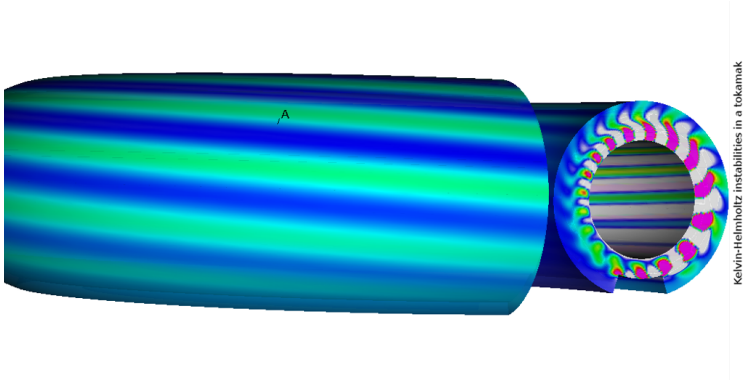
Contents

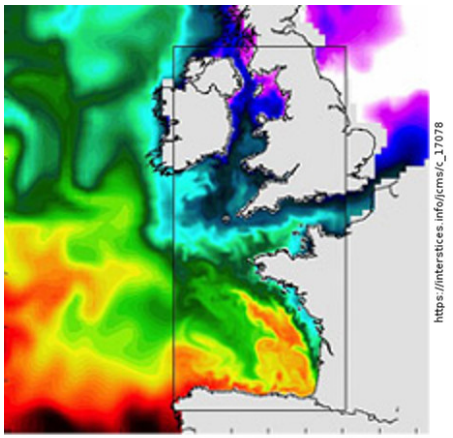
1

Introduction

Context

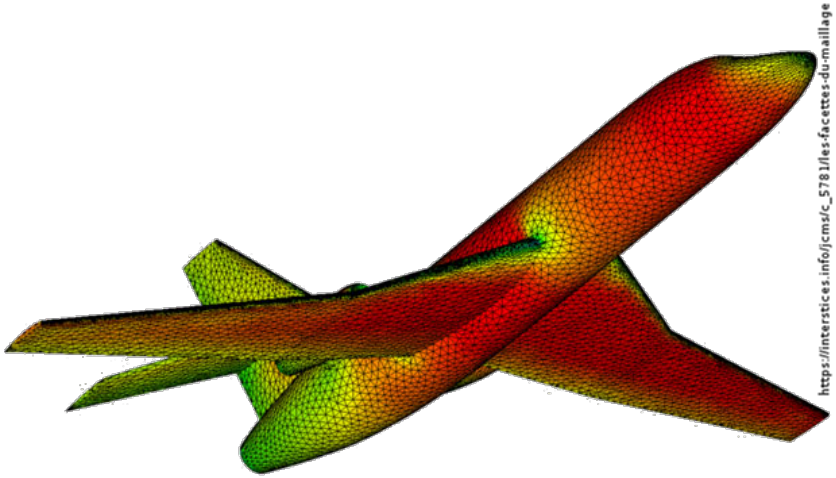
- Numerical simulations are needed in multiple domains including:
  - thermonuclear fusion
  - aeronautics
  - meteorology, ...
- Problems become bigger and more and more complex: numerical simulations cannot be run on a single workstation
  $\rightarrow$ Want of parallelized computations
  $\rightarrow$ Need to distribute data across the processors: domain decomposition

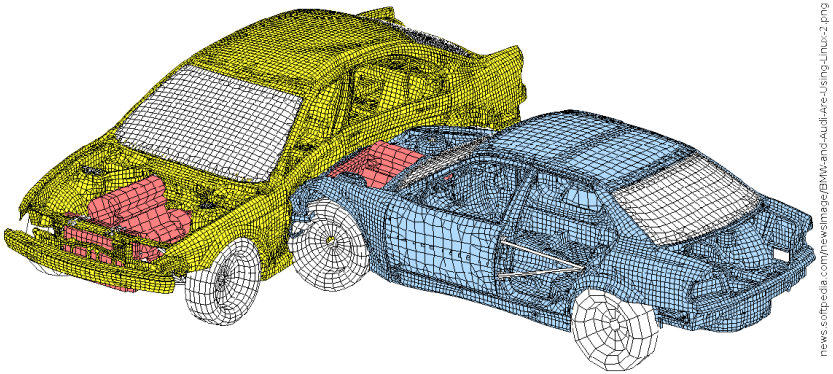




Numerical simulation

- Space discretization:
  - mesh
- Finite number of points on which values of the problem are computed, e.g.:
  - temperature
  - pressure
  - speed,. . .
- Solution precision depends on mesh quality:
  - need for remeshing



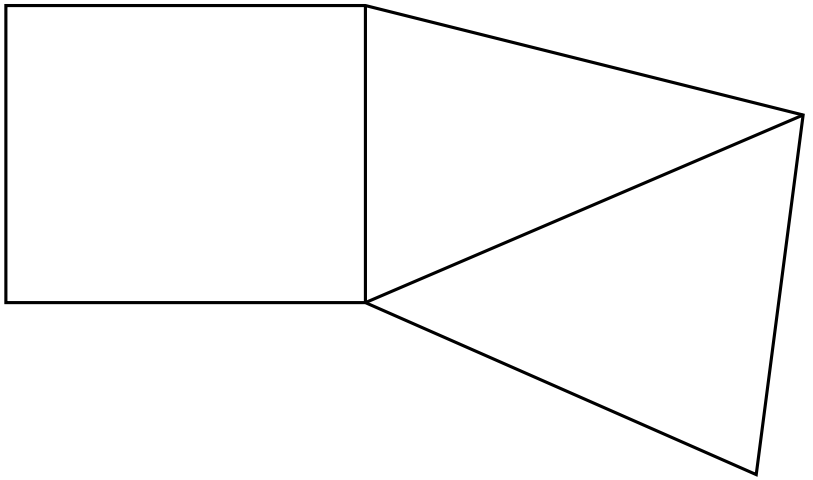


2

Data structures for parallel remeshing

Definitions

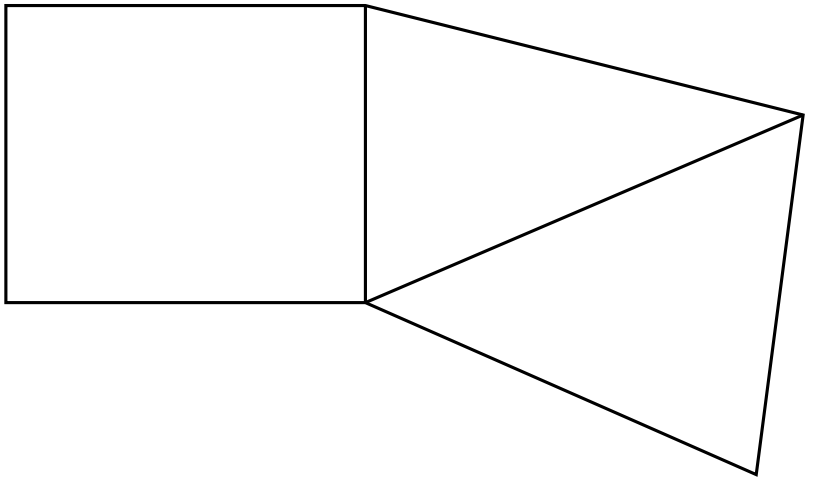
- **Mesh:**
  - **Node**
  - **Edge**
  - **Element**



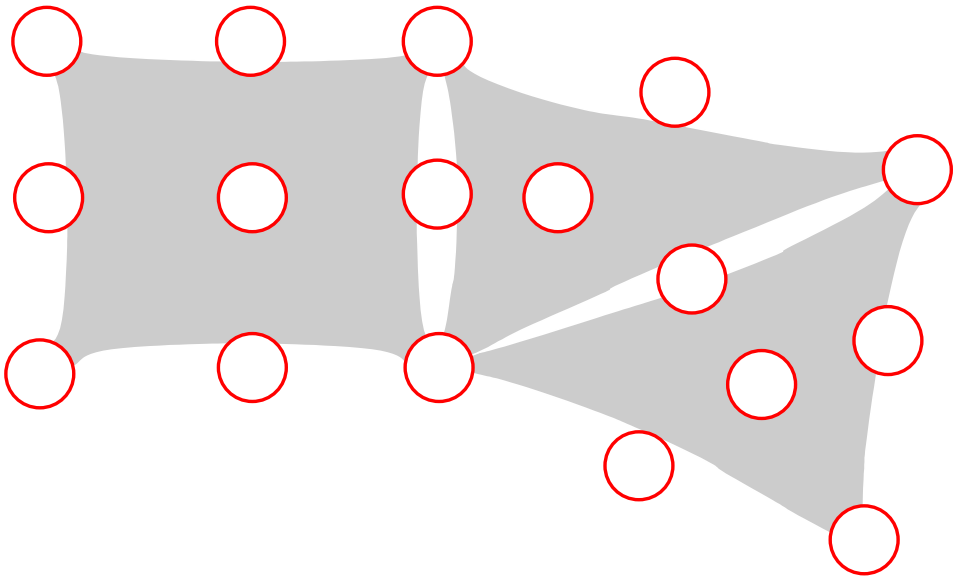
- **Mesh representation:**

Definitions

- **Mesh:**
  - **Node**
  - **Edge**
  - **Element**

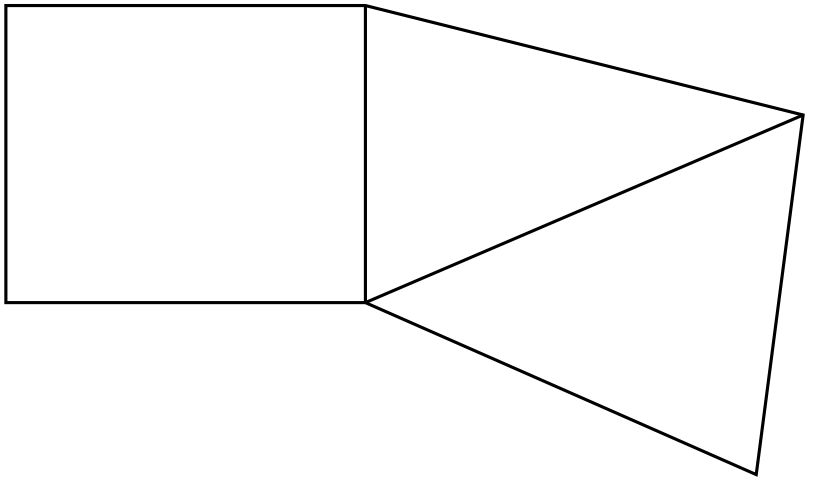


- **Mesh representation:**
  - **Vertex**

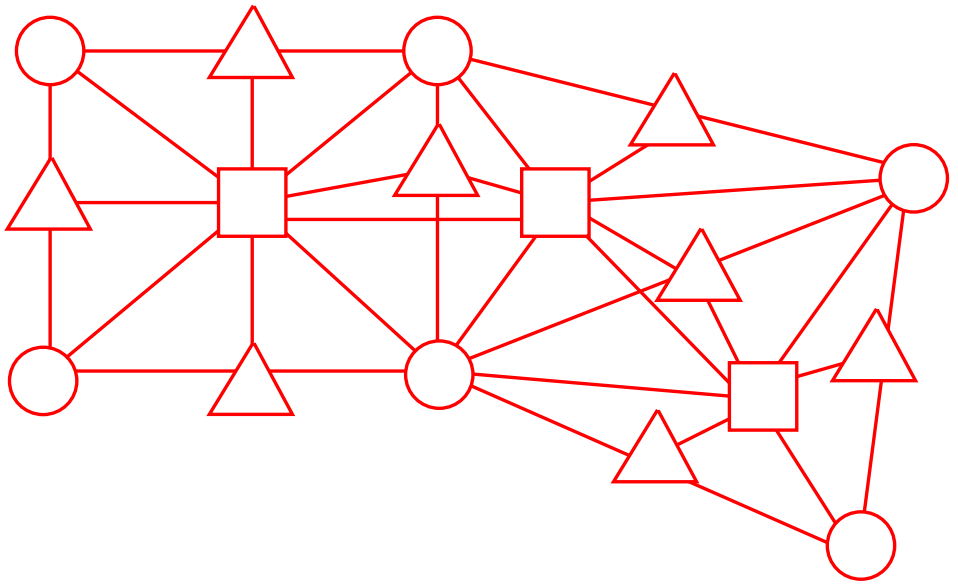


Definitions

- **Mesh:**
  - **Node**
  - **Edge**
  - **Element**

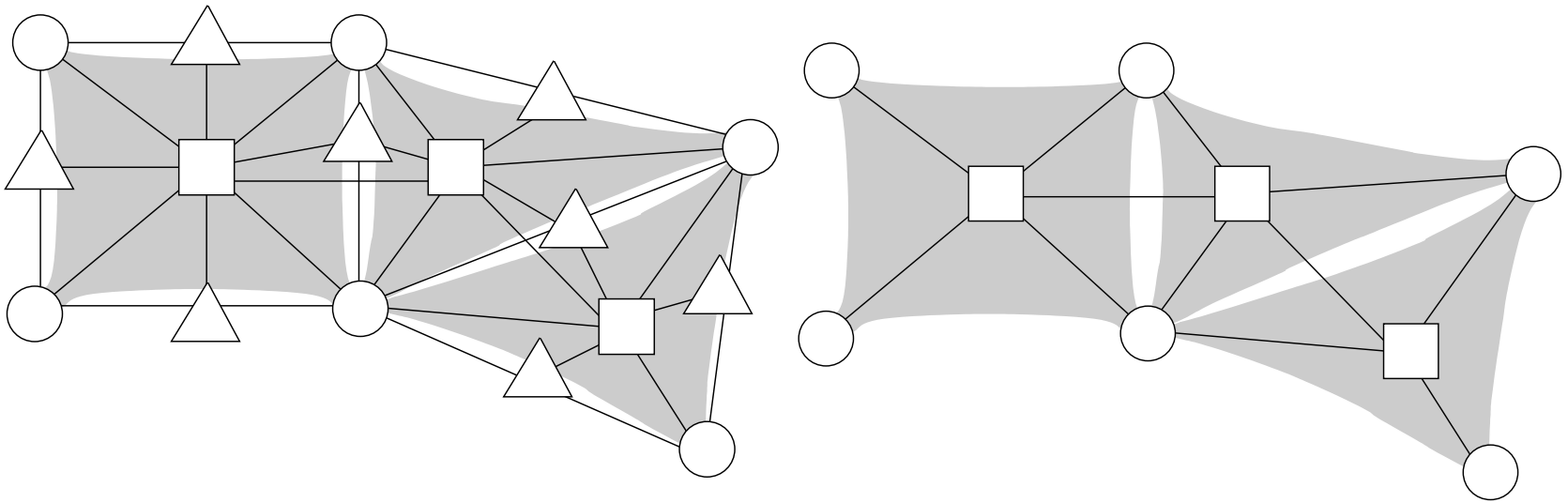


- **Mesh representation:**
  - **Vertex**
  - **Enriched graph**



Examples

- The same mesh can lead to different enriched graphs
  - Depending on the requirements of the numerical schemes



3

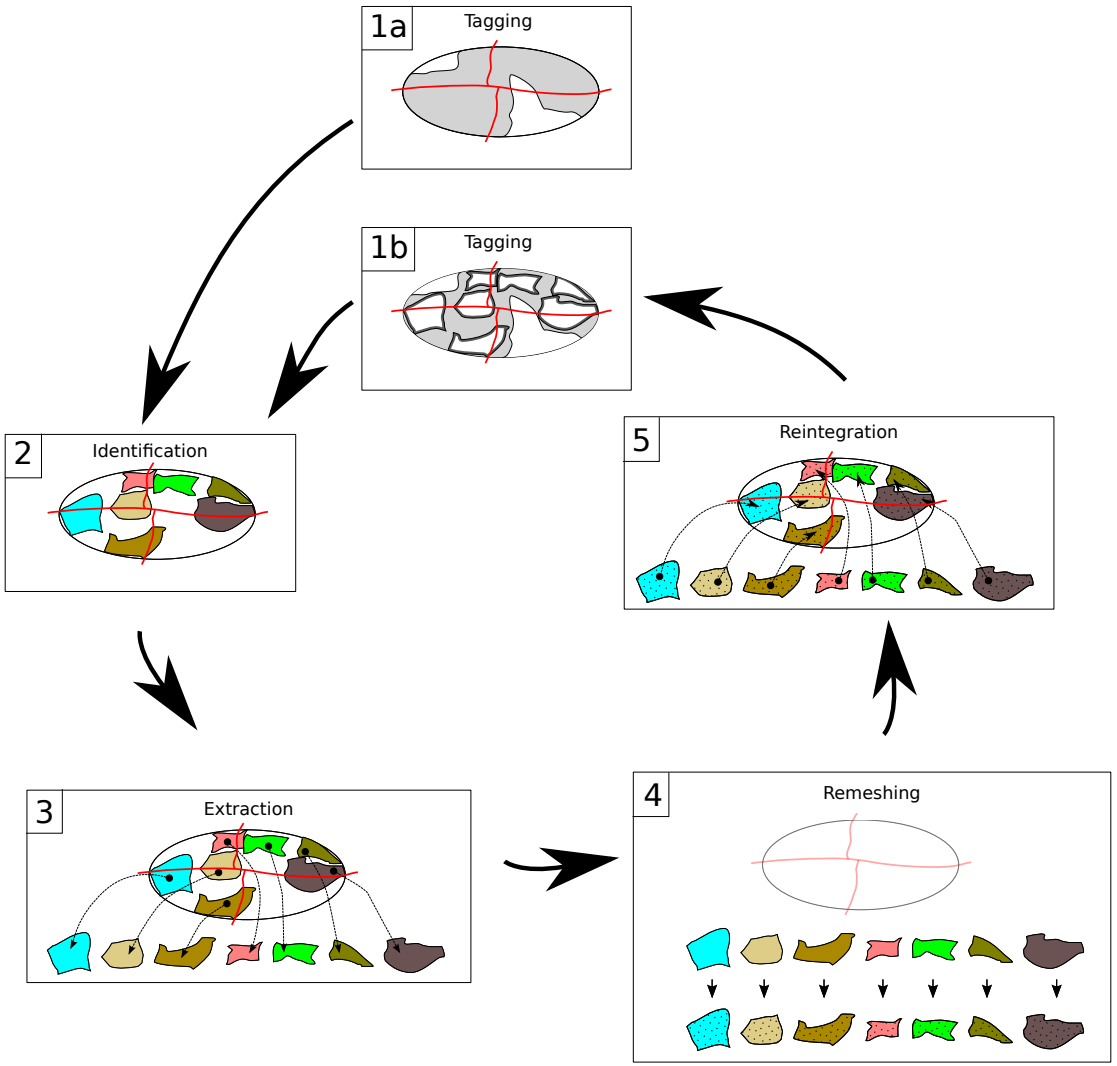
Parallel remeshing

Requirements

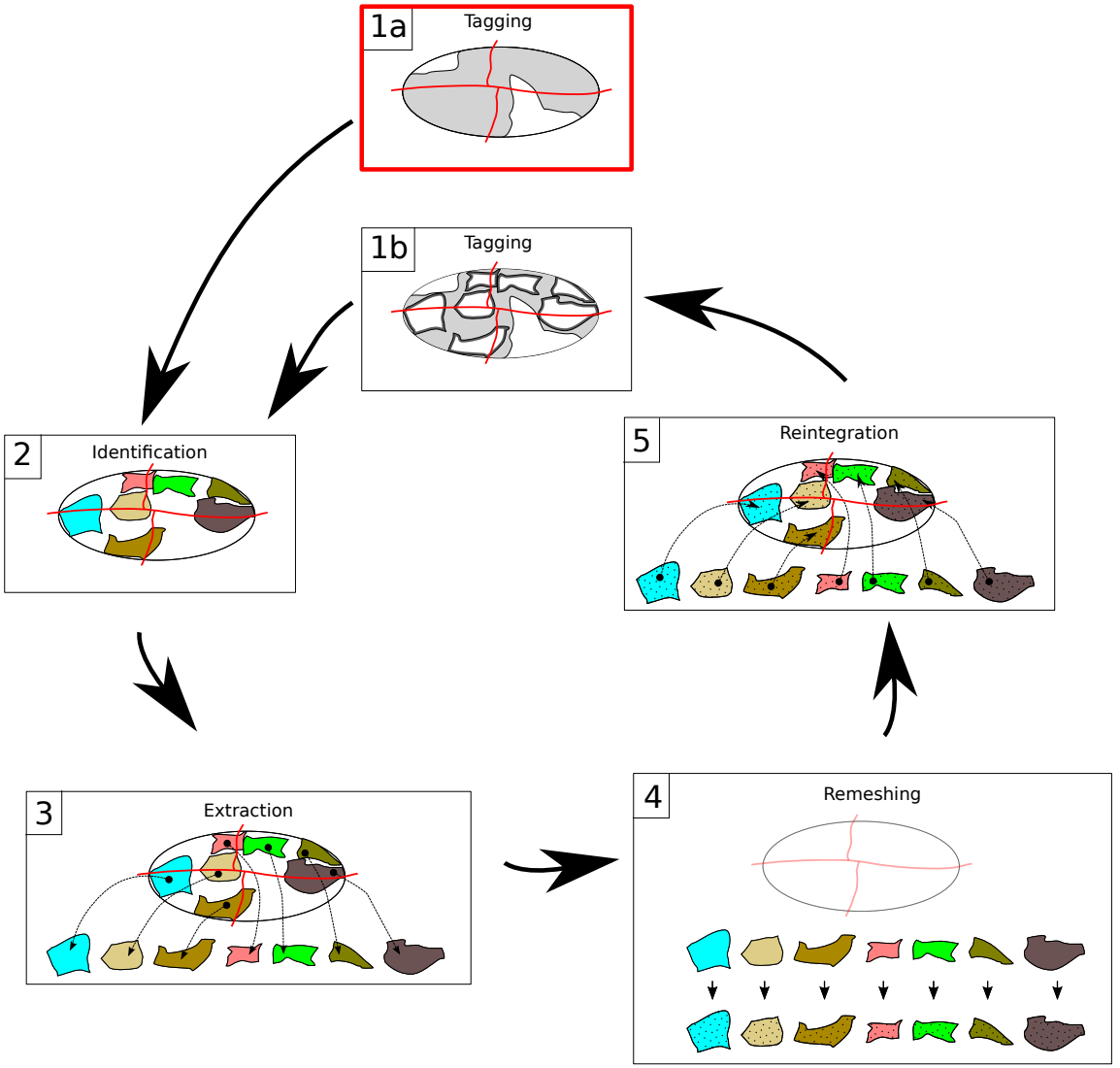
- How to remesh in parallel on distributed meshes?
  - Subdomain size could be greater than mesh size allowed by the sequential remesher
    $\rightarrow$ Need to identify in parallel non-overlapping zones of prescribed size / workload
  - Zone frontiers must be left unmodified so as to ease reintegration of remeshed zones into the distributed mesh
- How to determine which elements need to be remeshed?
  - Isotropic or anisotropic metric on vertices
  - Avoid elements of bad quality
- Which criteria on identified zones?
  - Measure zone skin size compared to zone volume:
    - compute the isoperimetric quotient of the zone
    - value between 0 and 1
    - 0: worst quotient possible
    - 1: optimal quotient, in the case of sphere and the corresponding ball

Global scheme:

Iterative process until all tagged elements are remeshed

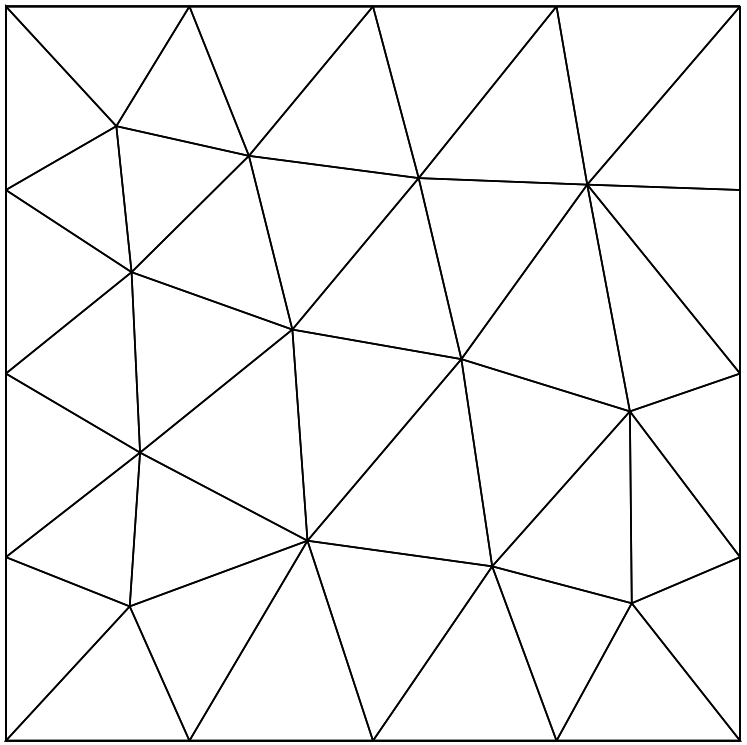


Tagging (1/3)

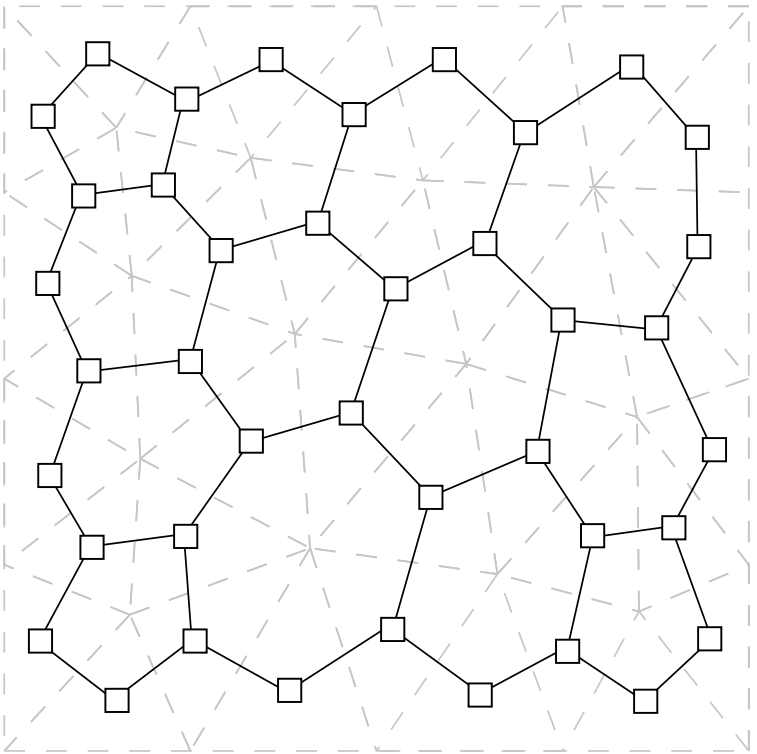

1a
Tagging
1b
Tagging
2
Identification
5
Reintegration
3
Extraction
4
Remeshing

Tagging (2/3)

- User-provided mesh:

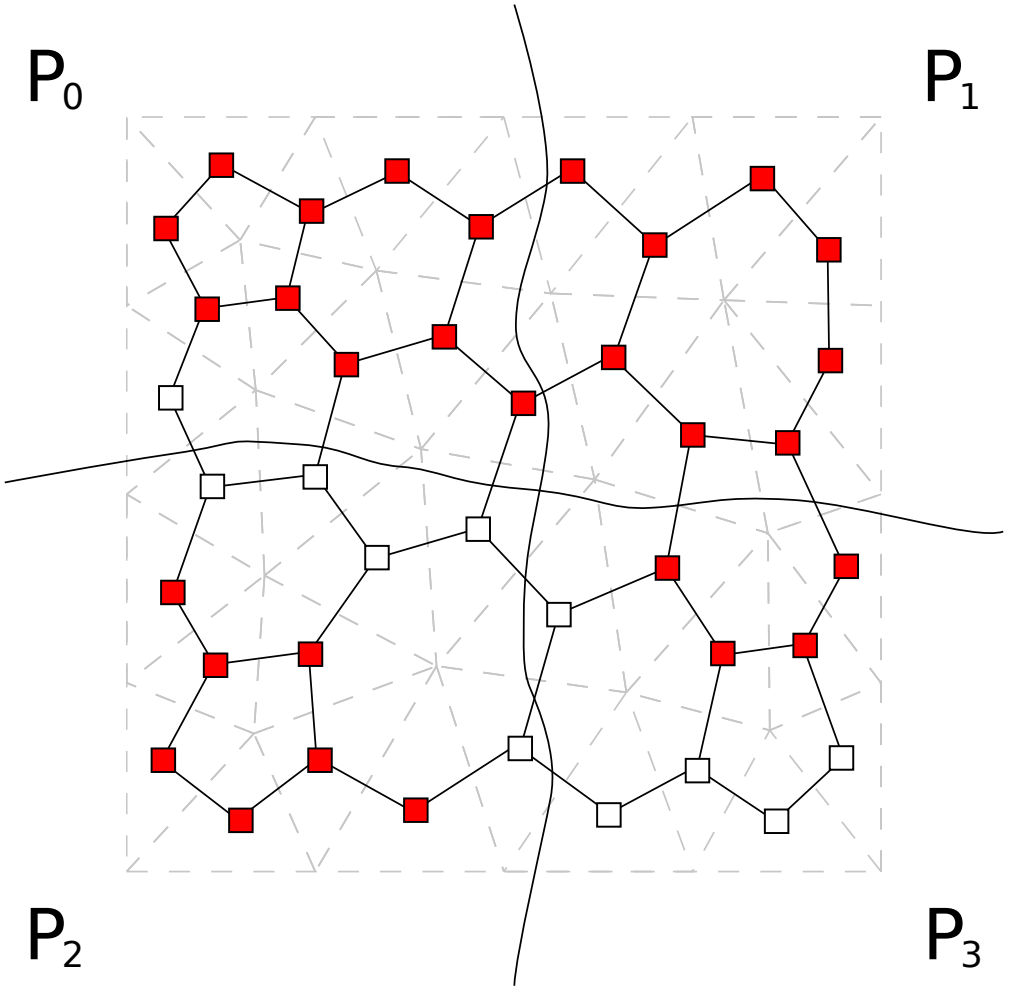


- Corresponding element graph:

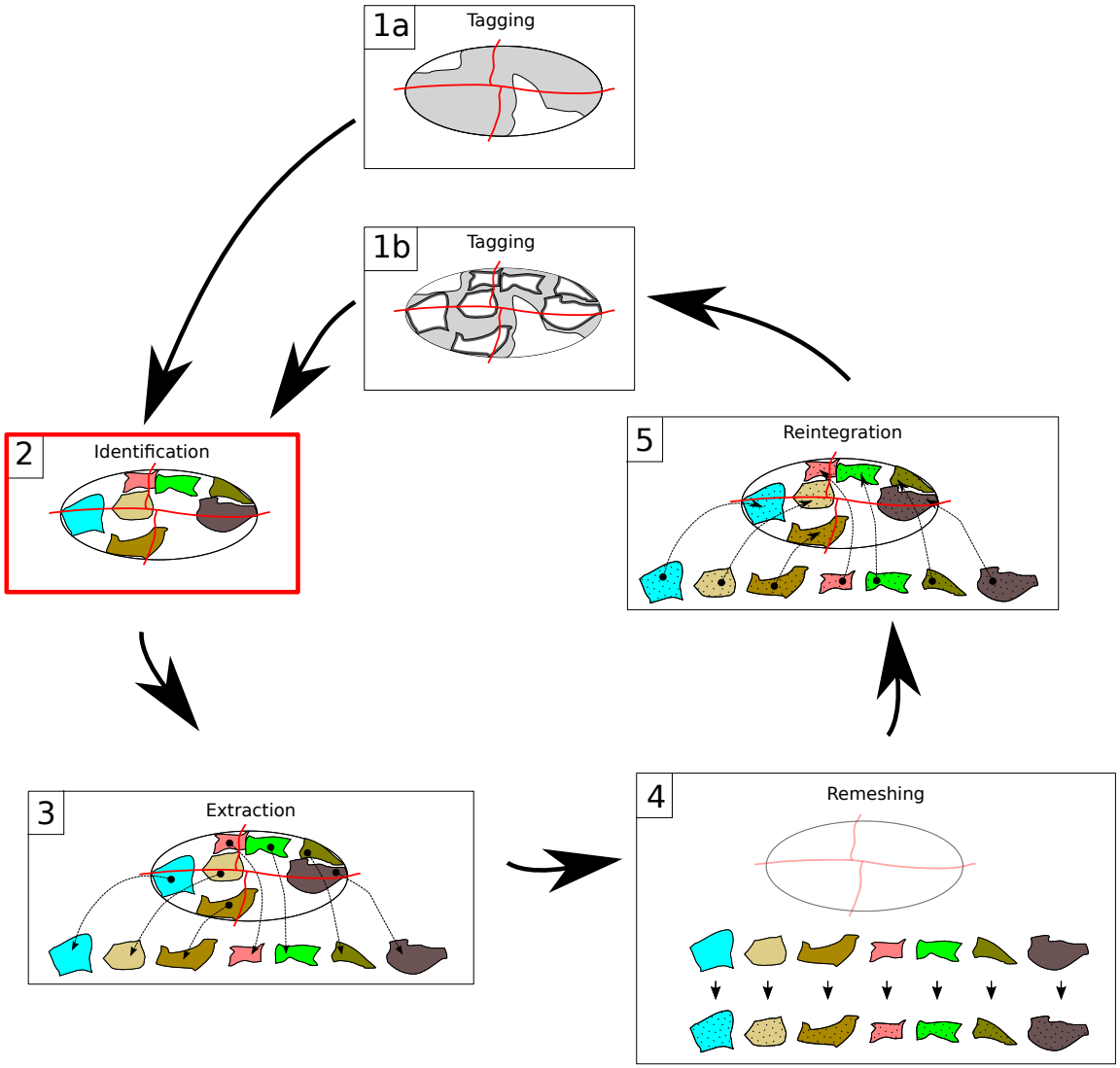


Tagging (3/3)

- Tag elements which need to be remeshed:

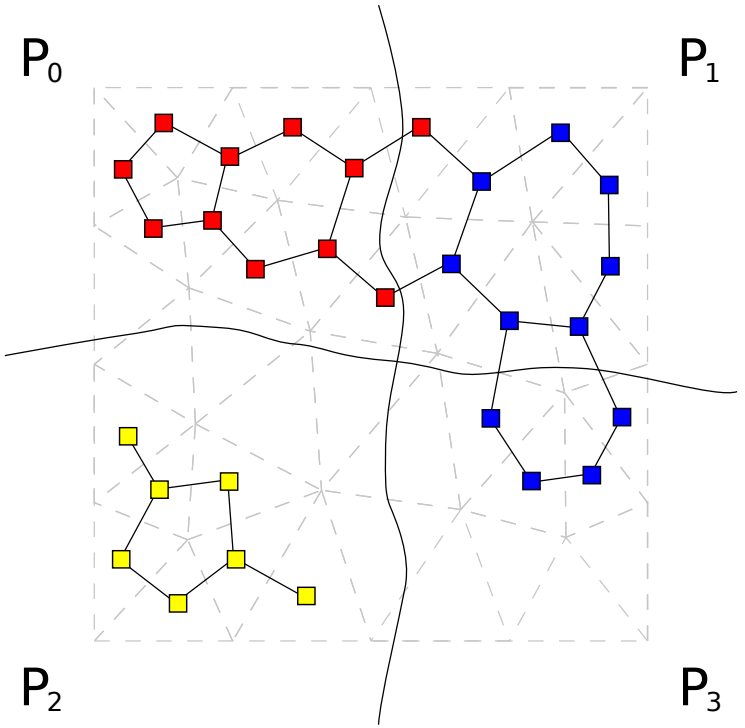


Identification (1/5)



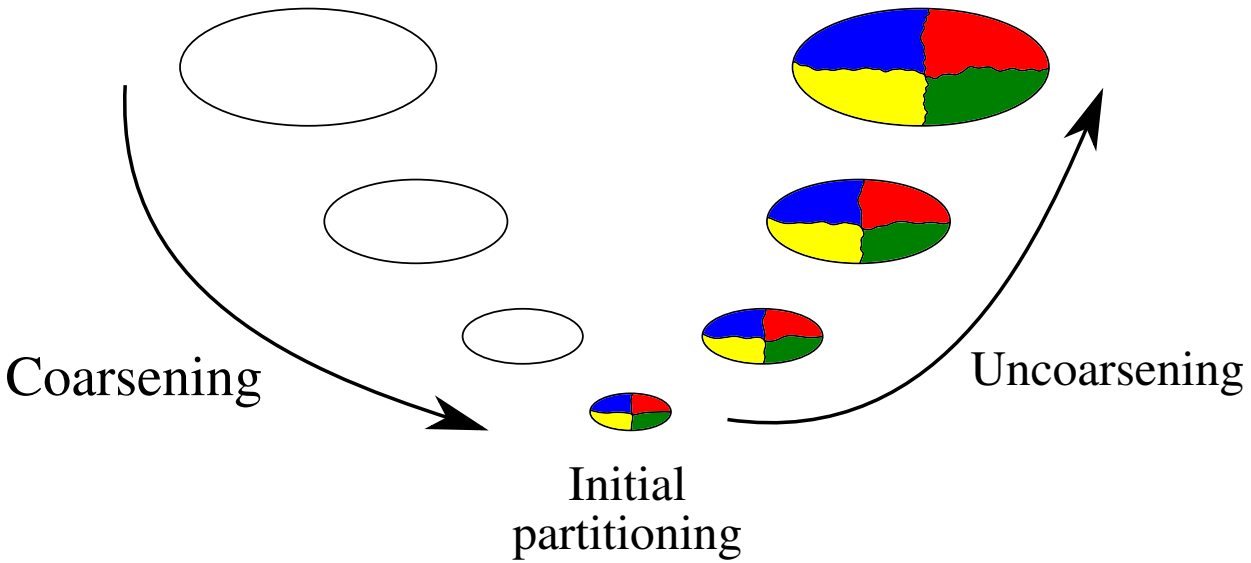
Identification (2/5)

- Identify non-overlapping zones in parallel:
  - each zone will be given to a sequential remesher
  - a weight is determined on each element according to the dedicated work of the remesher
    $\rightarrow$ Want of a good evaluation of the remesher work



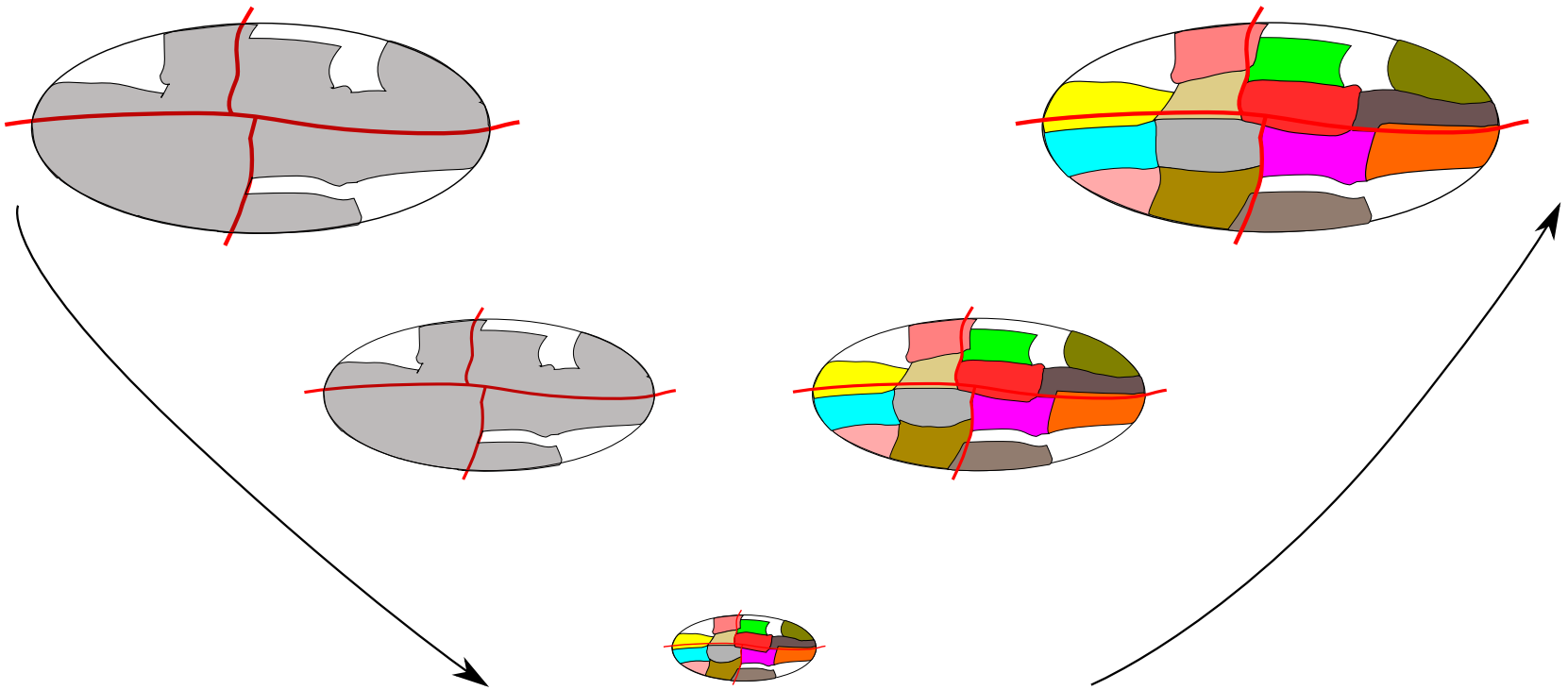
Identification (3/5)

- How to compute efficiently non-overlapping zones in parallel on a distributed mesh?
  - By using a graph multilevel scheme which provides very good partitions when combined to local optimization algorithms



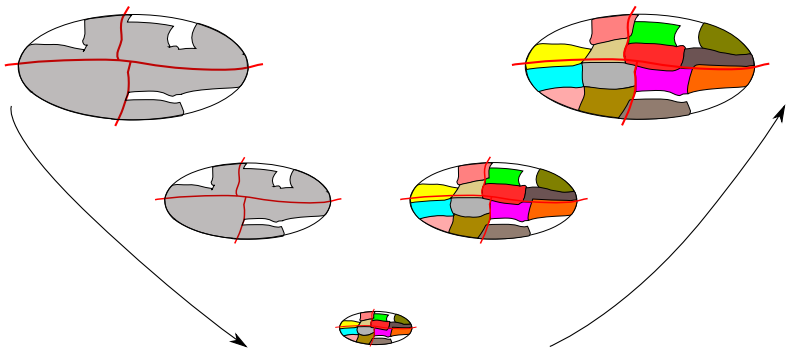
Identification (4/5)

- Algorithm: full-featured parallel partitioning
  - Local optimization is mandatory to obtain good isoperimetric quotients



Identification (4/5)

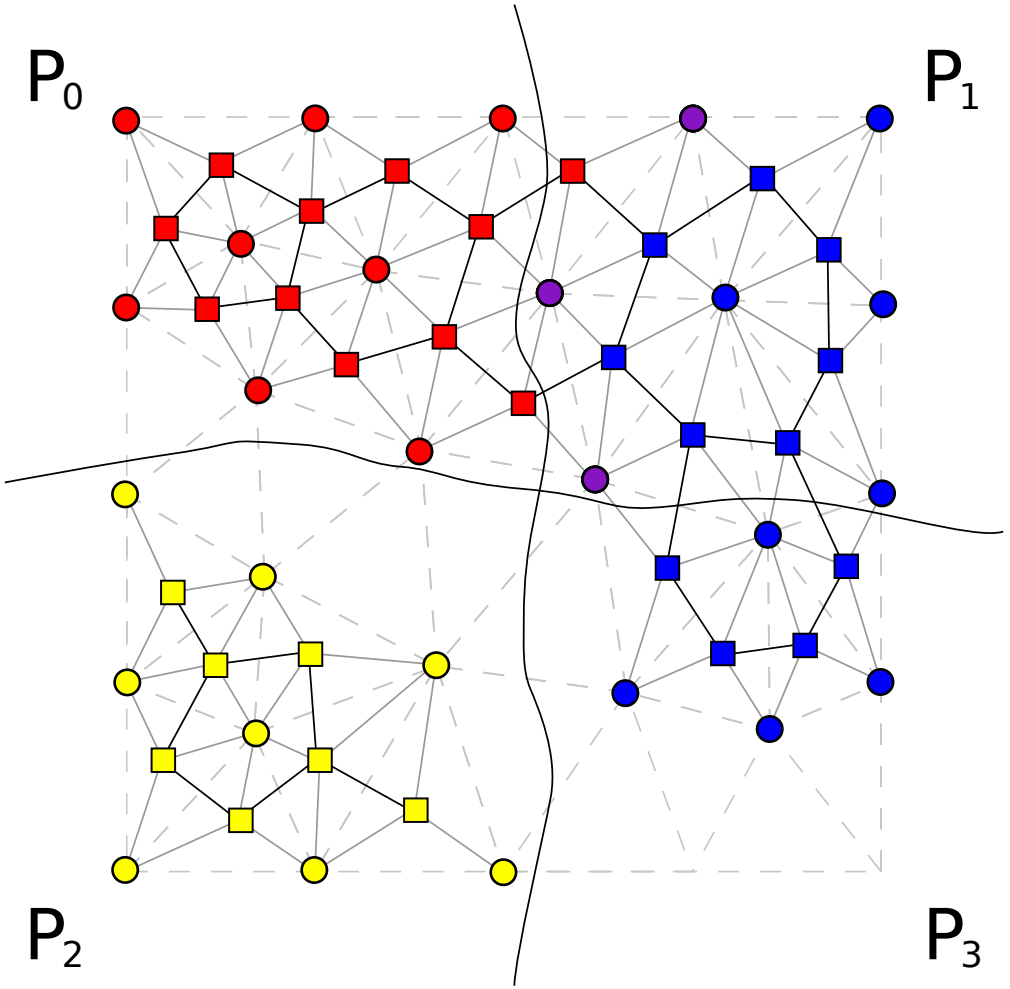
- Algorithm: full-featured parallel partitioning
  - Local optimization is mandatory to obtain good isoperimetric quotients



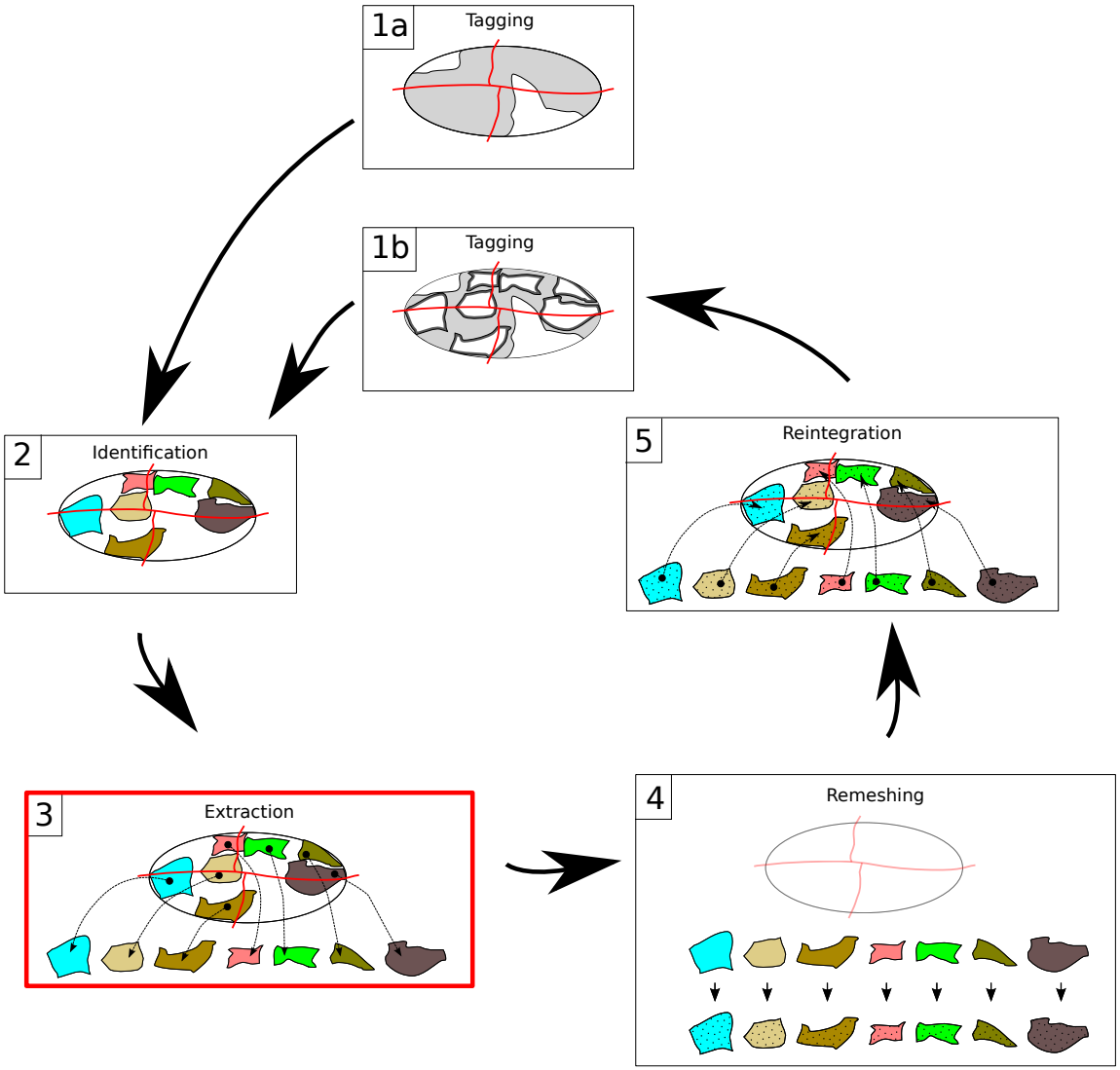
- Pros:
  - Smooth skin which gives good enough isoperimetric quotient
  - Number of zones does not depend on number of processors

Identification (5/5)

- Propagate zone color on other vertices through cells:

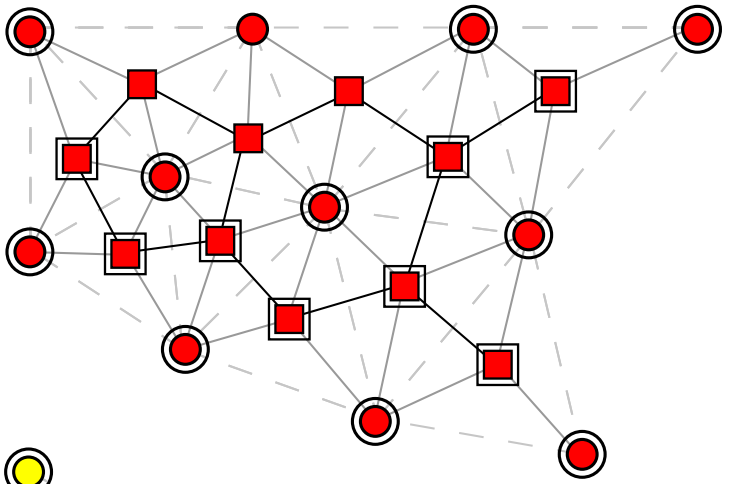


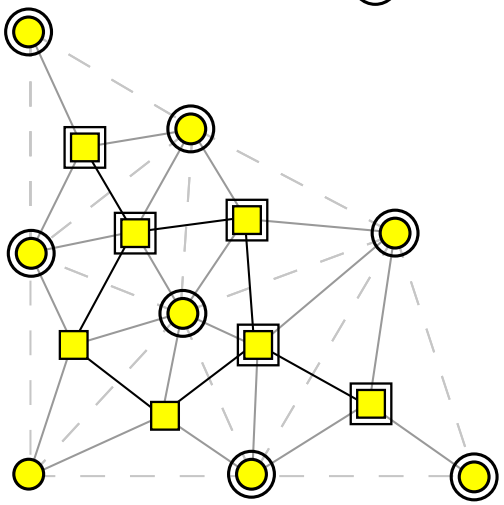
Extraction (1/3)

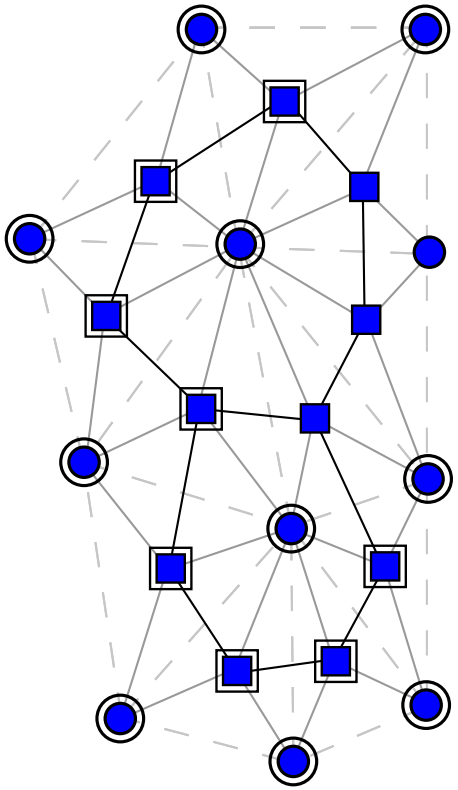


Extraction (2/3)

- Extract zones and distribute them throughout the processors:
  - zone skins will not be remeshed, except zone skins which correspond to mesh skin

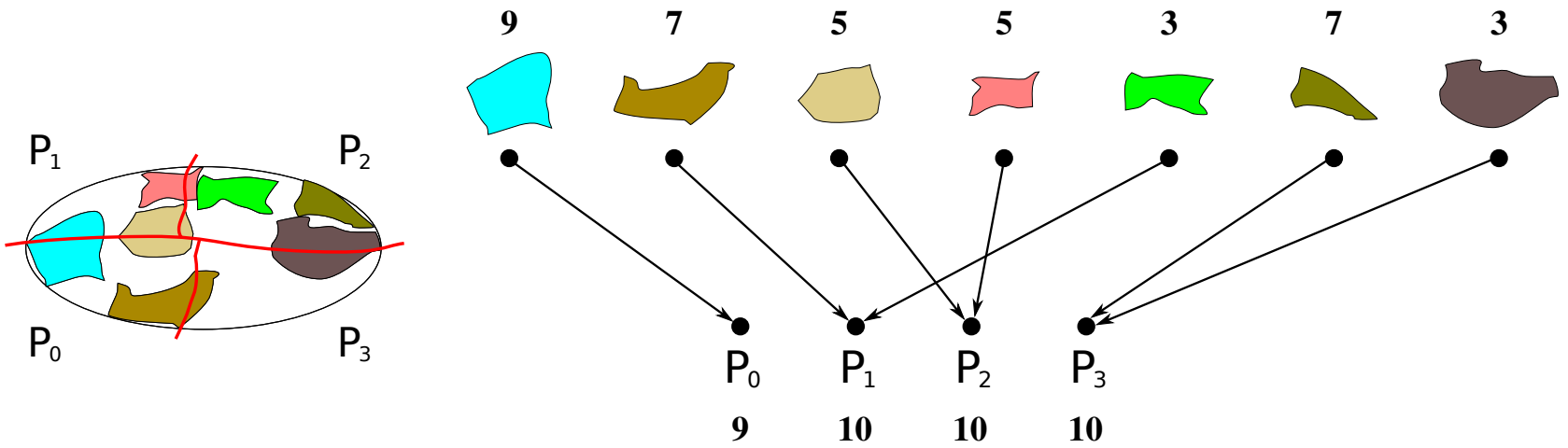






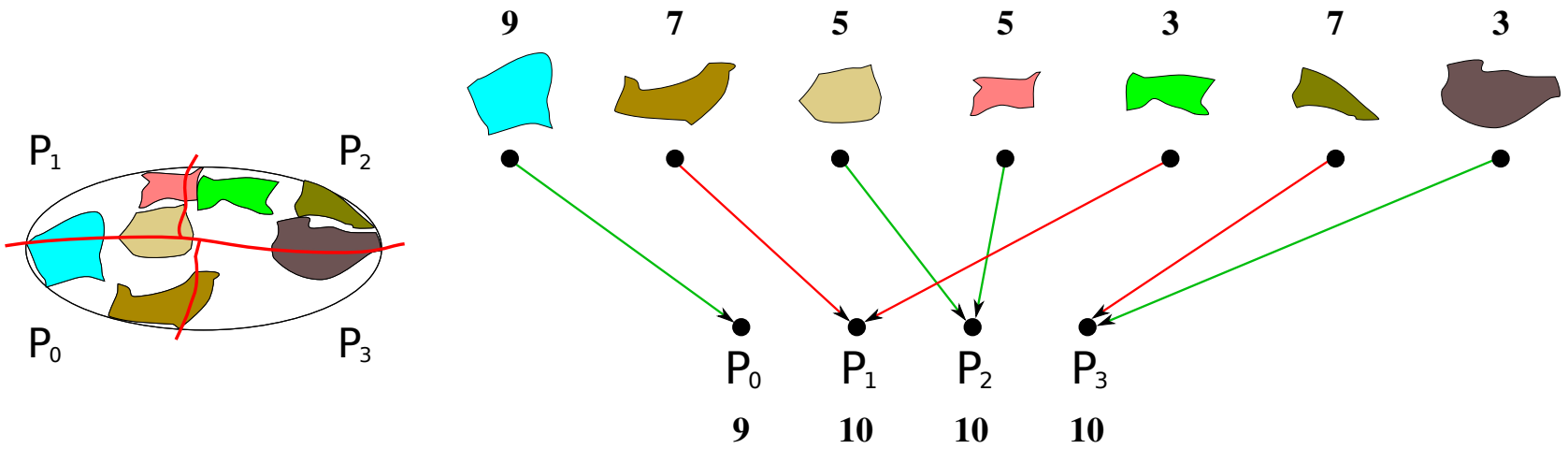
Extraction (3/3)

- Two criteria must be considered:
  - load-balance according to the remesher workload



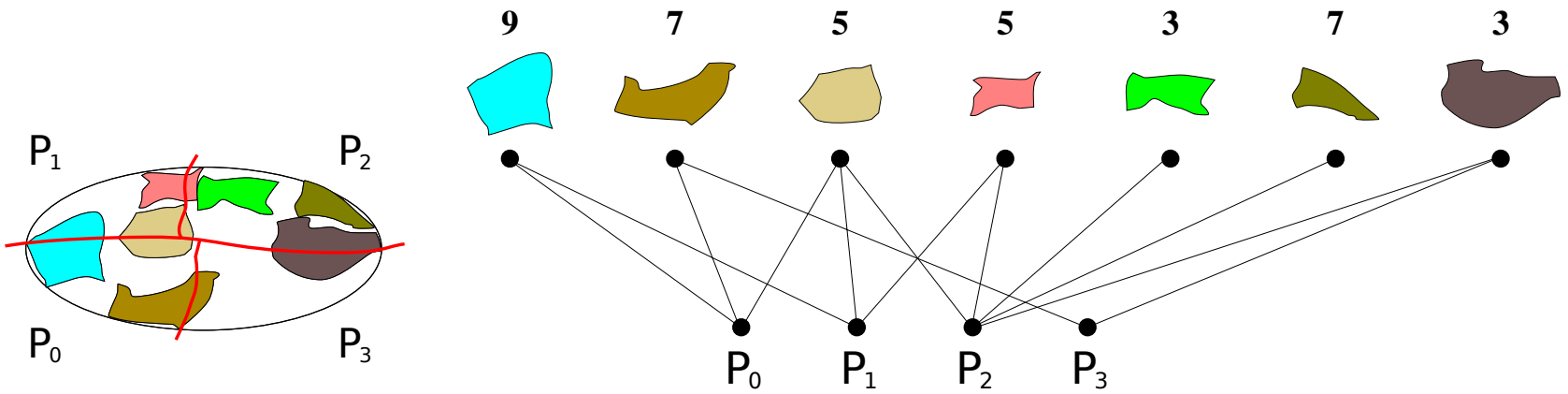
Extraction (3/3)

- Two criteria must be considered:
  - load-balance according to the remesher workload
  - minimize communication



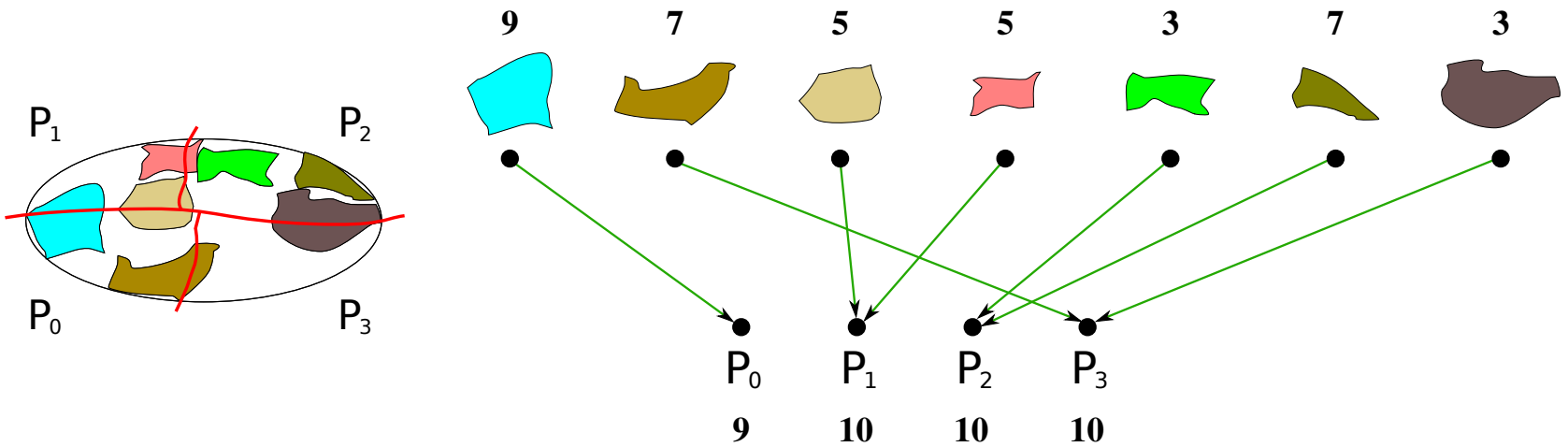
Extraction (3/3)

- Two criteria must be considered:
  - load-balance according to the remesher workload
  - minimize communication



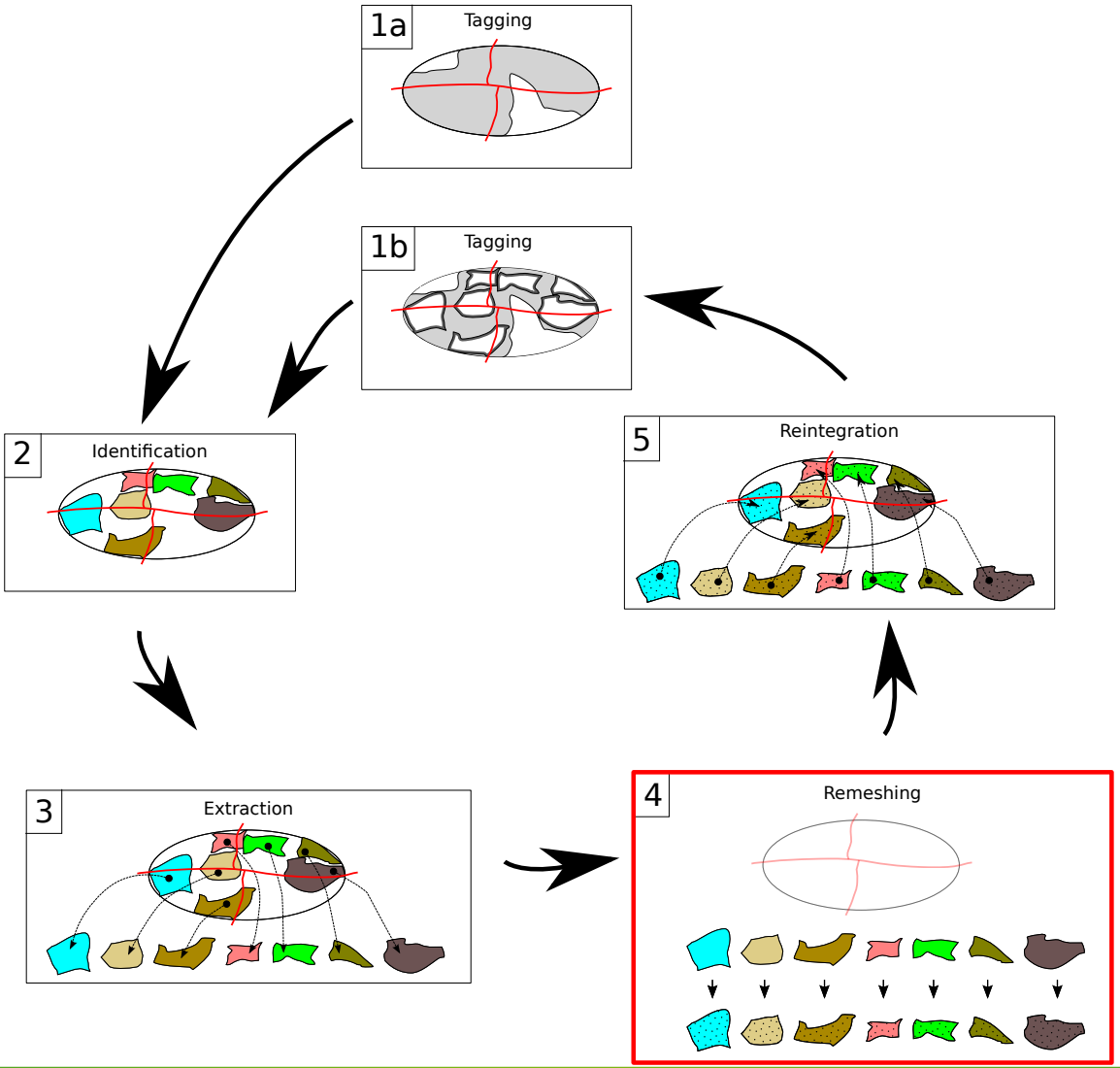
Extraction (3/3)

- Two criteria must be considered:
  - load-balance according to the remesher workload
  - minimize communication



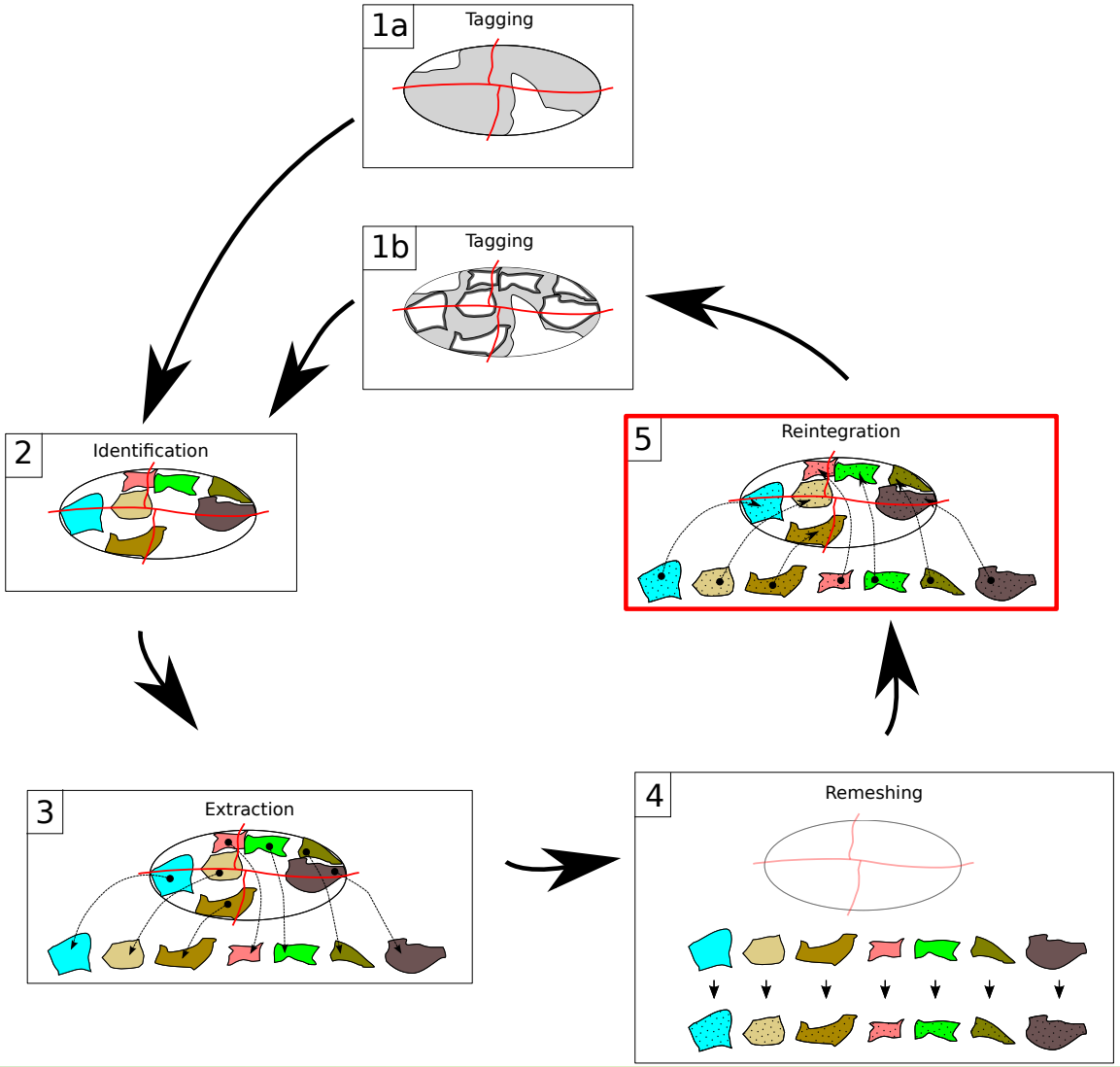
Remeshing

Remesh concurrently and sequentially each zone on separate processors:

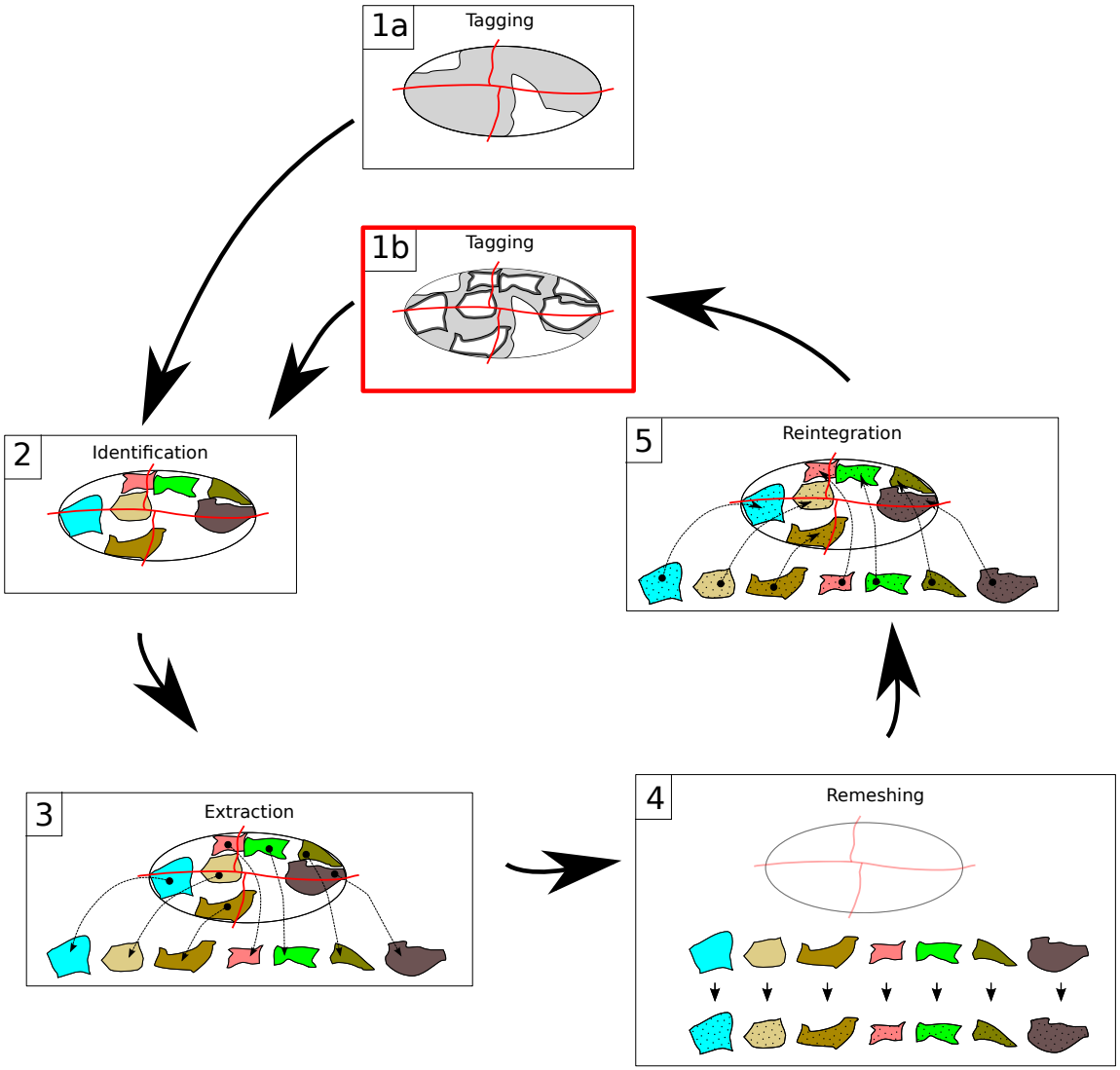


Reintegration

Fixed skin zone allows us to reintegrate zones in the distributed mesh:

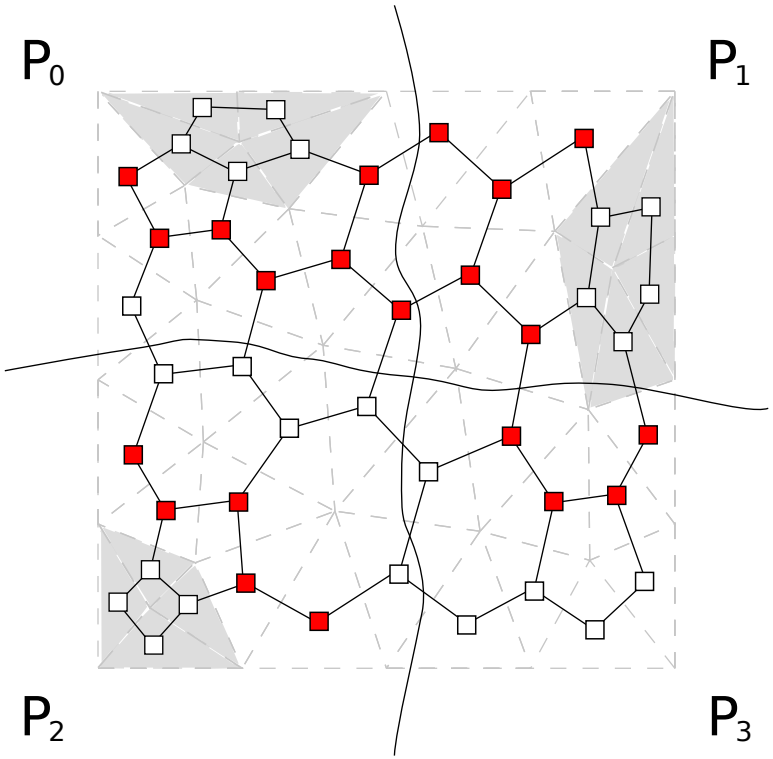


Tagging again (1/2)

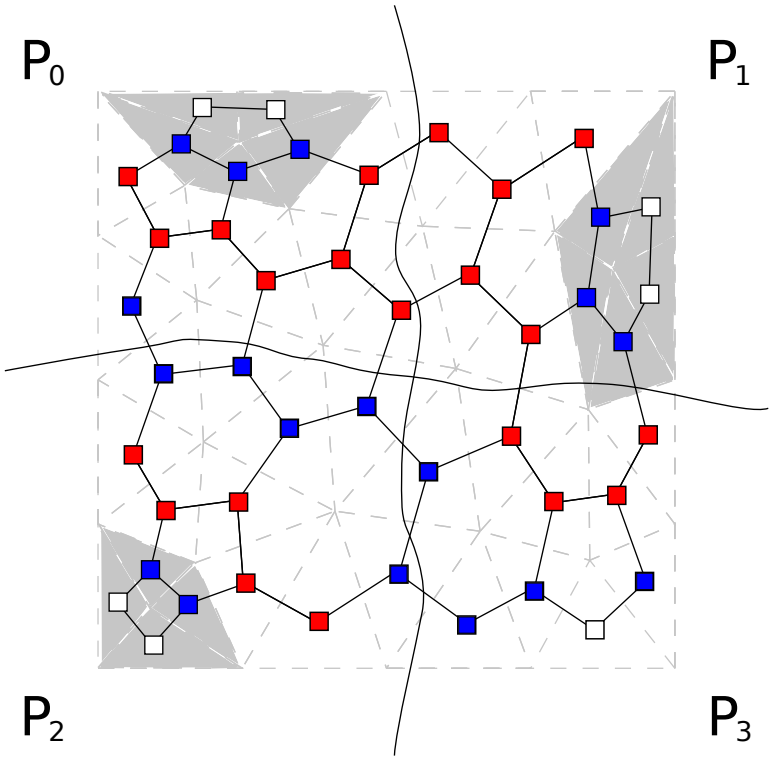


Tagging again (2/2)

- Remove tag on elements which are inside zones
- Yet a band from the skin is needed to remesh zone skin:
- Before tagging:



- After tagging:



4

Multigrid

Multigrid meshes

- Multigrid can be described as:
  - several distributed meshes
  - links between distributed meshes
- Distributed meshes are:
  - provided by the user
  - produced by recursively coarsening meshes by merging elements and removing nodes
- Partitioning on the coarsest mesh
- Propagation of the partition to finer levels
- Load-balancing each level
- Computing overlap for all the levels according to the coarsest mesh

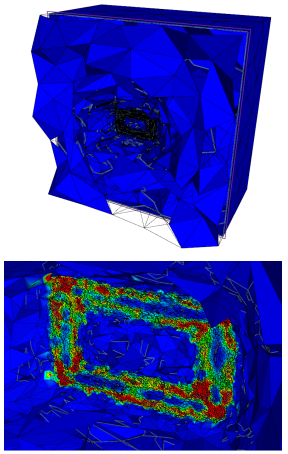
5

Experiments

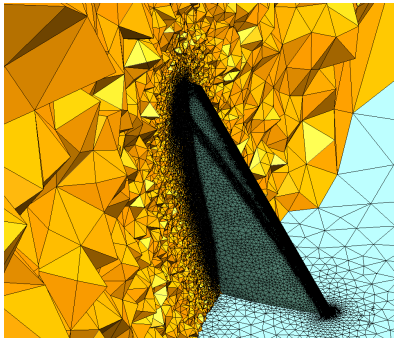
Parallel remeshing (1/5)

Isotropic mesh

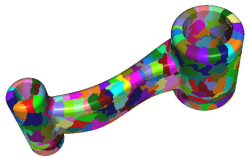
Anisotropic mesh



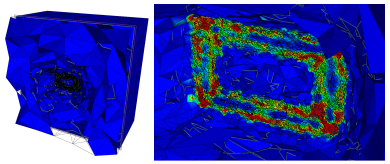
Remeshing according to a physical solution



Parallel remeshing (2/5): isotropic mesh

	PaMPA-MMG3D4 on 240 processors
Initial number of elements	27 044 943
Final number of elements	609 671 387
Elapsed time Elapsed time $\times$ number of PEs	00h34m59s 139h56m
Smallest edge length Largest edge length Worst element quality % element quality between 1 and 2 % edge length between 0.71 and 1.41	0.2911 8.3451 335.7041 98.92% 97.20%

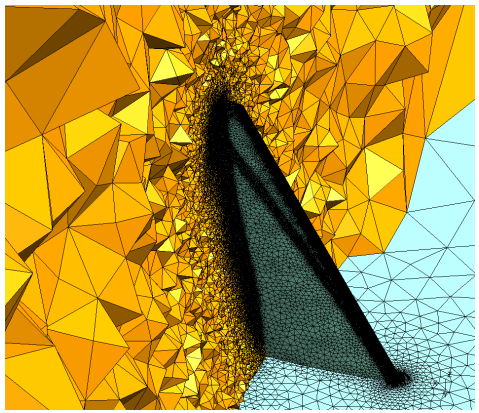
Parallel remeshing (3/5): anisotropic mesh



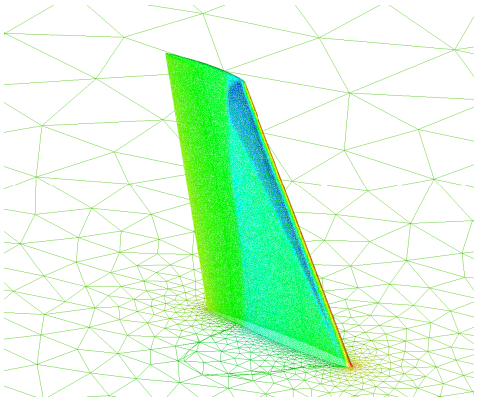
	PaMPA-MMG3D4 on 48 processors
Initial number of elements	715 791
Final number of elements	29 389 210
Elapsed time	00h34m
Elapsed time $\times$ number of PEs	27h12m
Smallest edge length	0.1116
Largest edge length	8.2191
Worst element quality	14.8259
% element quality between 1 and 2	99.61%
% edge length between 0.71 and 1.41	93.65%

Parallel remeshing (4/5): according to a physical solution

- Mesh adaptation within adaptation loop:
  - Solve transsonic Eularian flow arround a M6 wing
  - Discretization using a residual distribution scheme
- Mesh adaptation guided by a metric:
  - Metric based on an *a posteriori* error estimate of the interpolation error

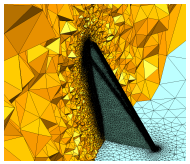


Volume cut



Density field

Parallel remeshing (5/5): according to a physical solution

	PaMPA-MMG3D5 on 5 processors	MMG3D5 on 1 processor
Initial number of elements	570 775	
Final number of elements	5 089 972	5 132 259
Elapsed time	00h07m	00h29m
Elapsed time $\times$ number of PEs	00h34m	00h29m
Smallest edge length	0.0422	0.2092
Largest edge length	2.4416	2.4416
Worst element quality	26.42	9.12
% element quality greater than 0.5	99.66%	99.65%
% edge length between 0.71 and 1.41	96.62%	95.67%

6

Conclusion

Conclusion

- PaMPA is dedicated to distributed meshes
- We have devised an efficient scheme for parallel remeshing of very large meshes, which can be coupled with any sequential remesher
- We can achieve the same quality as sequential remeshers
- Release of version 1.0
  - Available soon from Inria Gforge
  - Licensed under GPL

Upcoming features

- Mesh definition with a grammar
- Face orientation and displacement
- Parallel I/O with HDF5
- Parallel mesh adaptation scalability

Future of PaMPA

- Now:
  - Looking for companies willing to participate to a consortium to foster the industrial development of PaMPA
- Then:
  - Create a start-up company providing software edition and service around PaMPA
- Project team:
  - Cédric Lachat
  - François Pellegrini

Future of PaMPA

- Now:
  - Looking for companies willing to participate to a consortium to foster the industrial development of PaMPA
- Then:
  - Create a start-up company providing software edition and service around PaMPA
- Project team:
  - Cédric Lachat
  - François Pellegrini

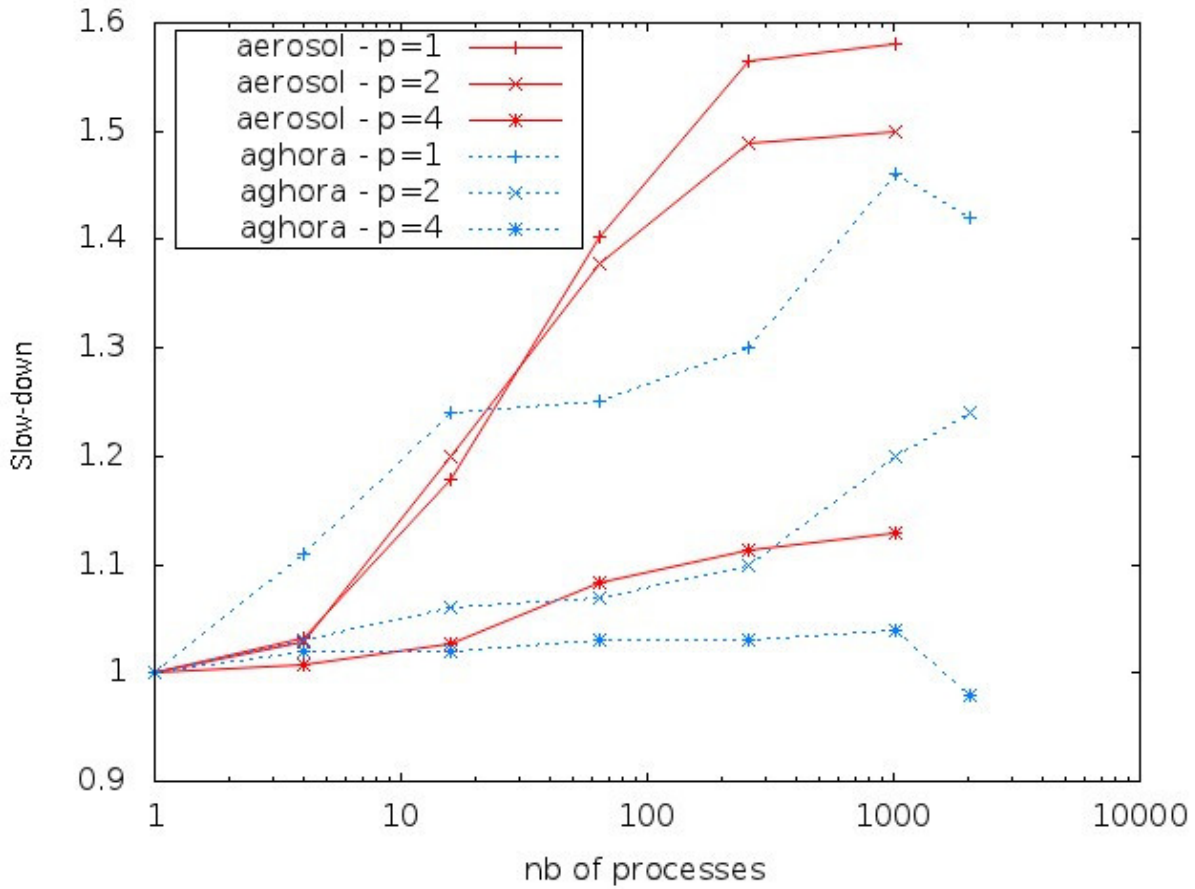
Thank you for your attention!

Functional test case (1/3)

- Our work implemented in a library: PaMPA
- PaMPA integrated in another library: Aerosol (continuous and discontinuous finite elements library on hybrid meshes)
- Comparison of Aerosol with the Aghora library (developed by ONERA) in the Yee vortex test case:
  - implemented into Aerosol by BACCHUS and CAGIRE teams (not including me)
  - isentropic vortex in a 3D uniform and inviscid flow
  - 3D discretization with hexahedra on the unit square $[0, 1]^3$ with periodic boundary conditions
  - discontinuous Galerkin method
  - explicit Runge-Kutta time stepping

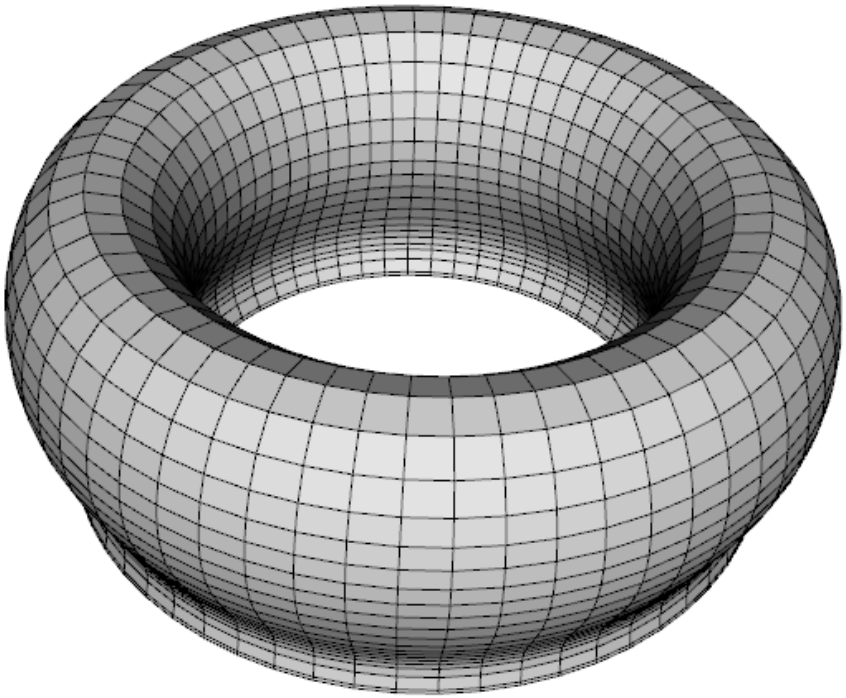
Functional test case (2/3)

- Comparison of the weak scalability
- Varying the polynomial degree *p*:



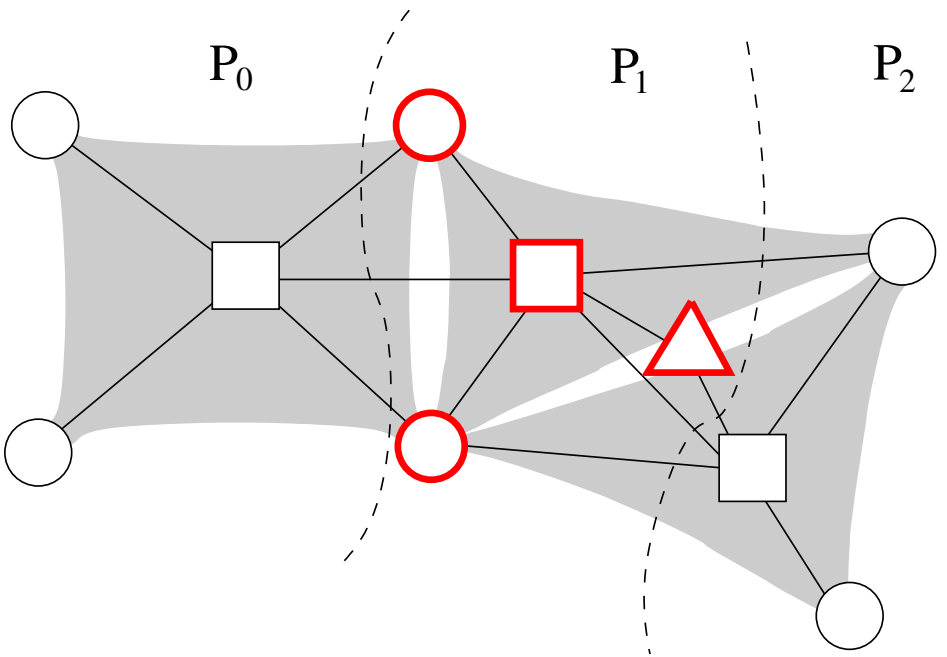
Functional test case (3/3)

- Anisotropic propagation of heat in a tokamak
- Unstationary anisotropic diffusion
- Torus mesh composed of prisms
- Discontinuous Galerkin method
- Implicit scheme
  - Linear system solved with PETsC
- Integrated by CAGIRE team



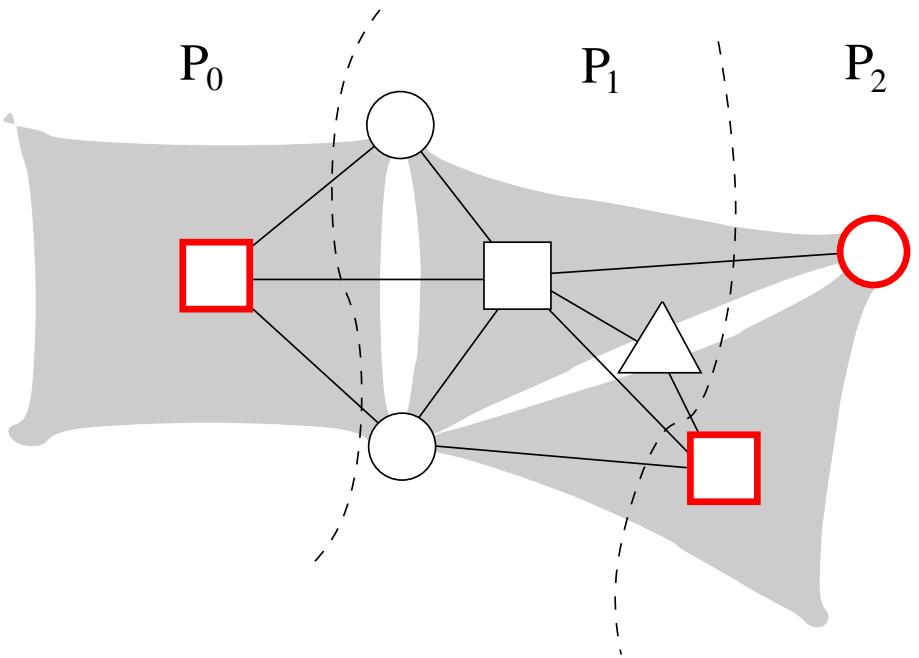
Definitions

- Vertices, locally on P_1:
  - **Local**
  - **Halo**
  - **Overlap**
- Vertices, between P_0 and P_1:
  - **Frontier**
- Vertices, globally:
  - **Internal**



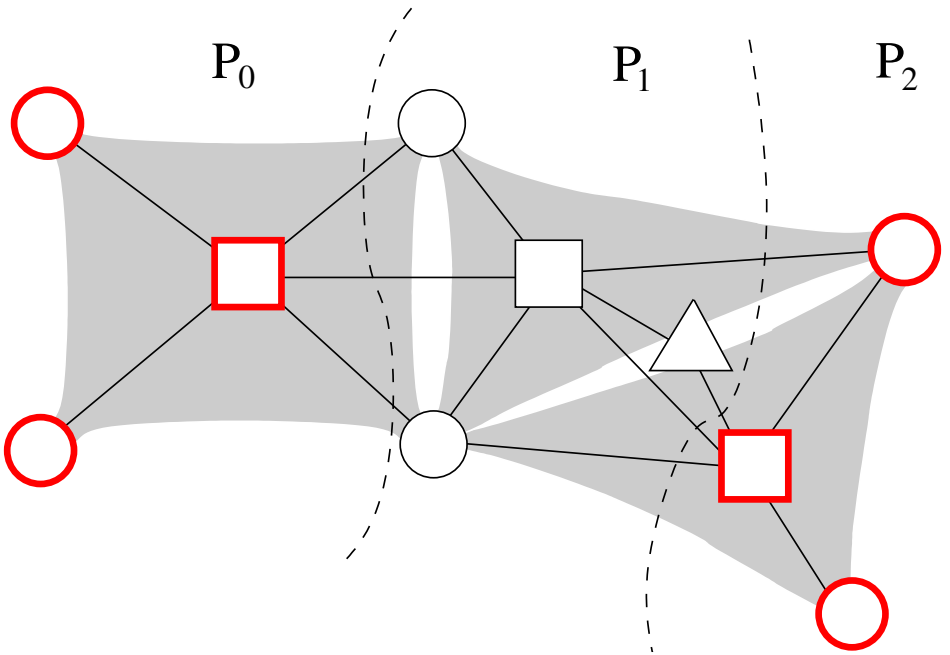
Definitions

- Vertices, locally on P_1:
  - **Local**
  - **Halo**
  - **Overlap**
- Vertices, between P_0 and P_1:
  - **Frontier**
- Vertices, globally:
  - **Internal**



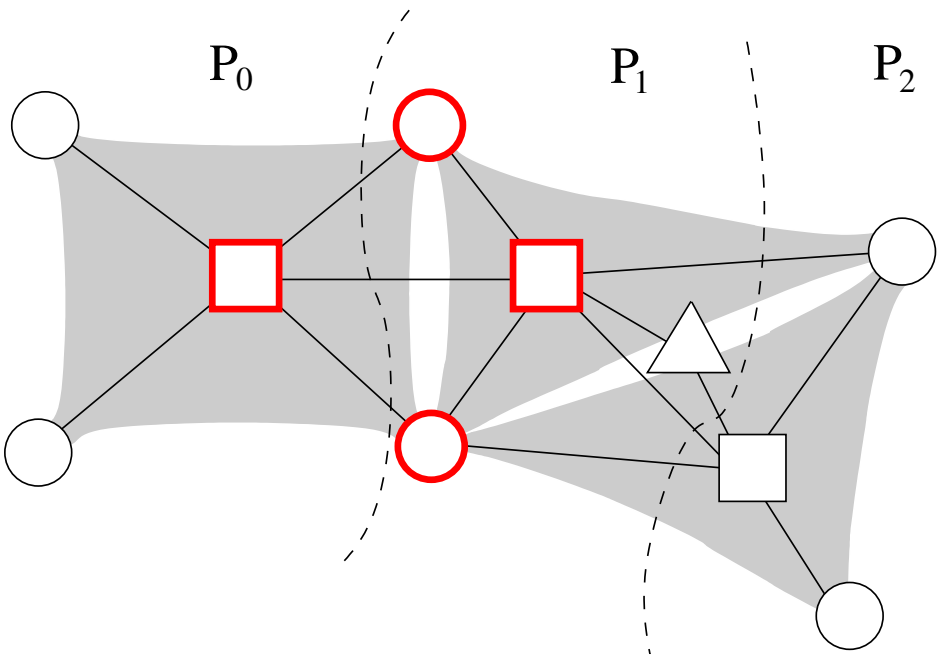
Definitions

- Vertices, locally on P_1:
  - **Local**
  - **Halo**
  - **Overlap**
- Vertices, between P_0 and P_1:
  - **Frontier**
- Vertices, globally:
  - **Internal**



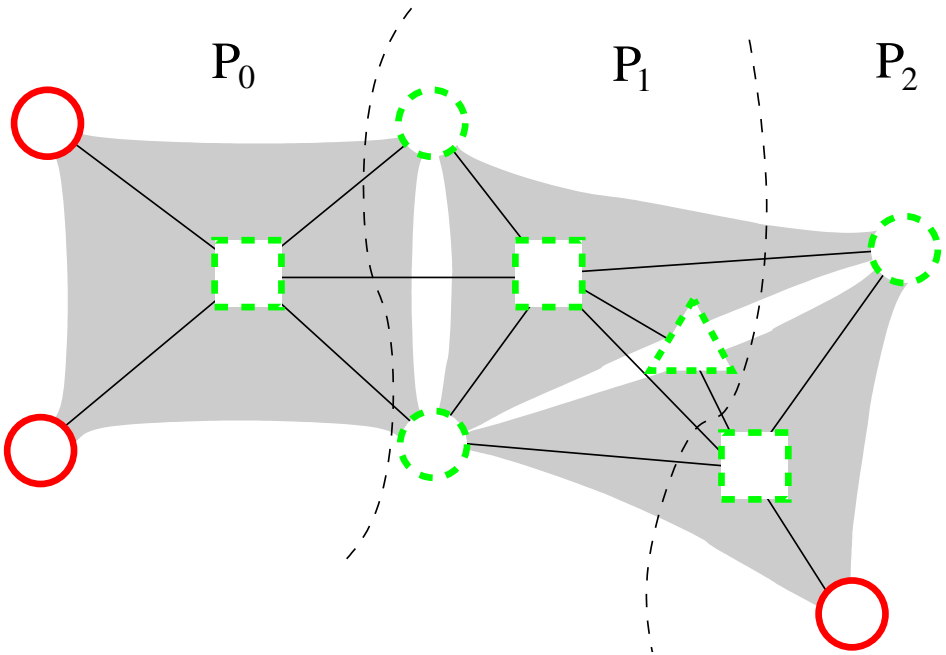
Definitions

- Vertices, locally on P_1:
  - **Local**
  - **Halo**
  - **Overlap**
- Vertices, between P_0 and P_1:
  - **Frontier**
- Vertices, globally:
  - **Internal**



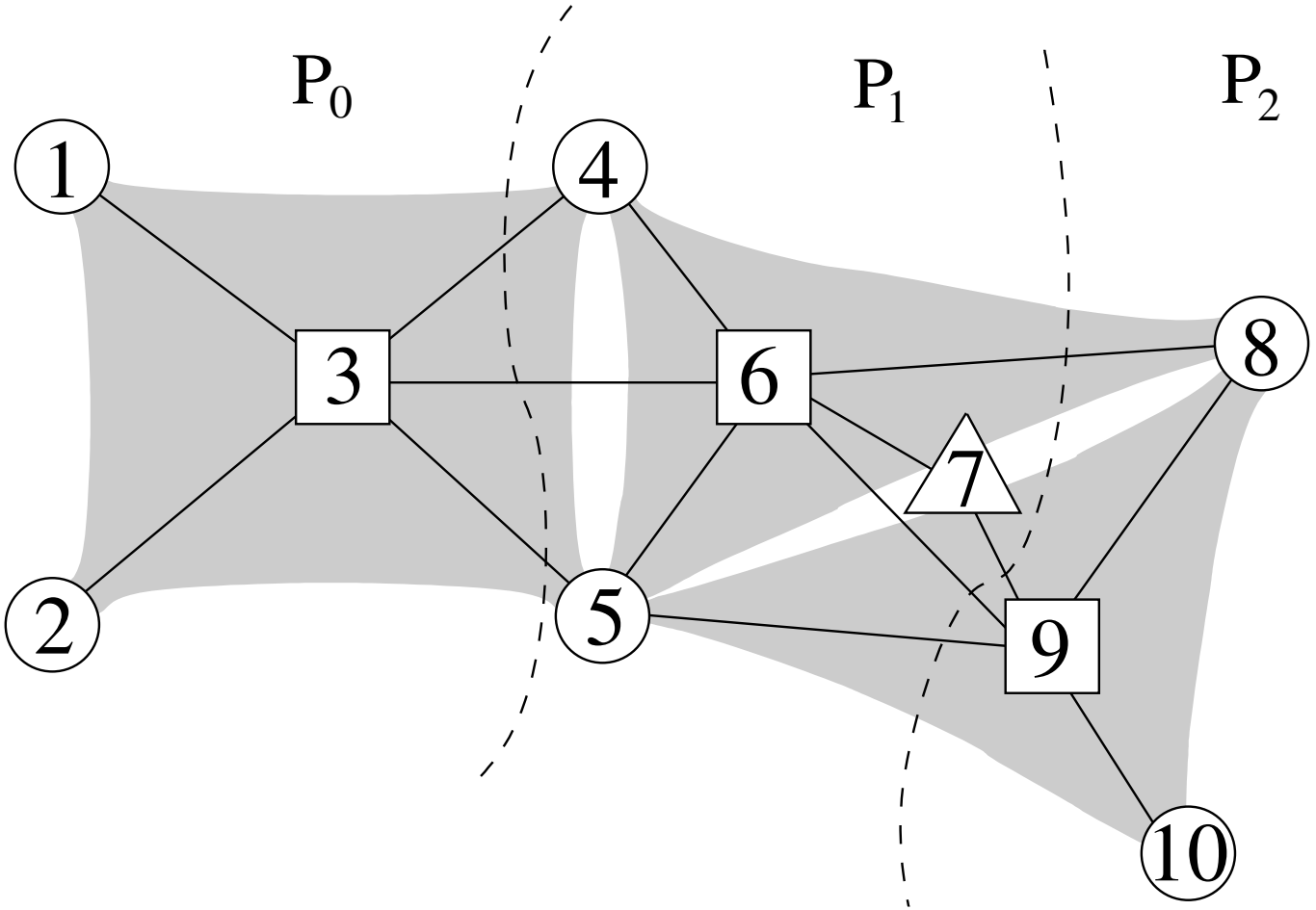
Definitions

- Vertices, locally on P_1:
  - **Local**
  - **Halo**
  - **Overlap**
- Vertices, between P_0 and P_1:
  - **Frontier**
- Vertices, globally:
  - **Internal**



Global numbering

- Necessary to describe the mesh in its entirety



Local numbering

- Local indices are mandatory to address local data arrays
- For all local vertices on every subdomain, vertices are indexed per entity
- Local indices are extended to index halo vertices as well, e.g. with respect to processor P_1: