AeroSol

Solver for CFD problem for modern architecture

INTRODUCTION

Many CFD platforms around the world.

- One element support: Triangle, Tetrahedron, etc.
- One dimension support: 2D, 3D.
- Solution: Low or high order
- Language..

Each develop its solver and only a few share.

$\Rightarrow$ **We want to achieve genericity, collaboration, and performance.**

Context

Software Architecture

Distributed Memory level

Shared Memory level

Conclusion, Perspectives

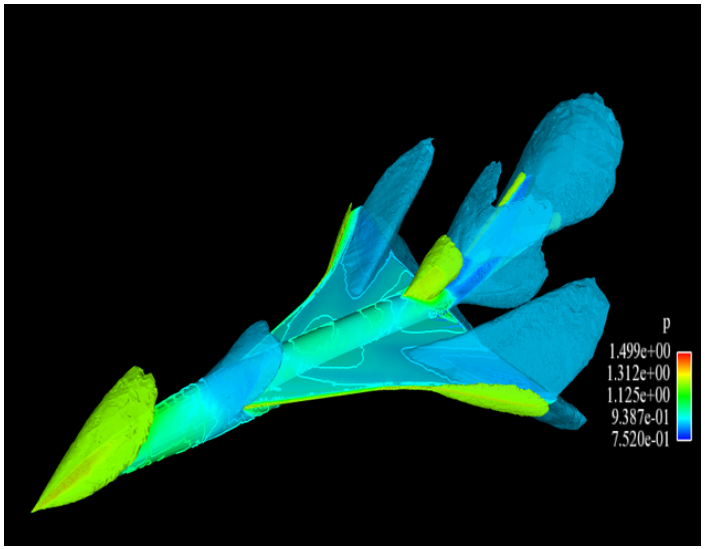
1 Context

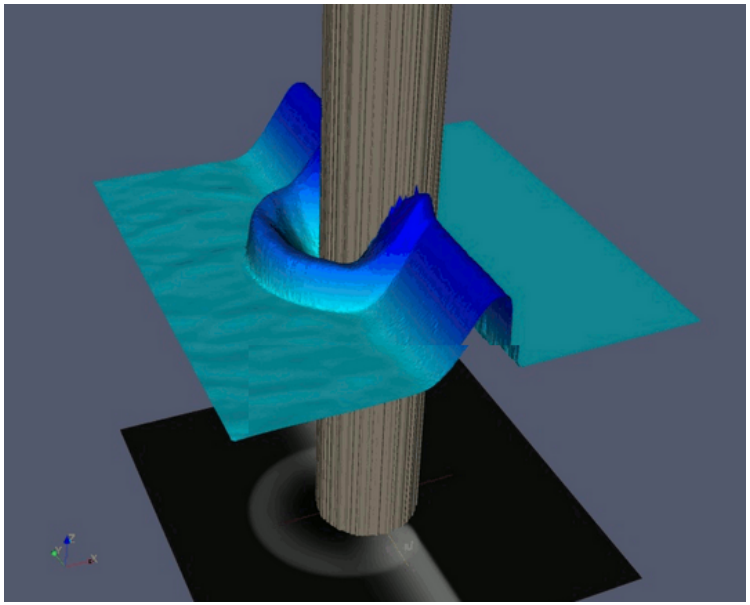
The story begins...

Bacchus

Bacchus, INRIA project:

- HPC tools: `Scotch` , `MMG3D` , `PaMPA` .
- CFD solvers: `FluidBox` , `RealFluid` , `AeroSol` .

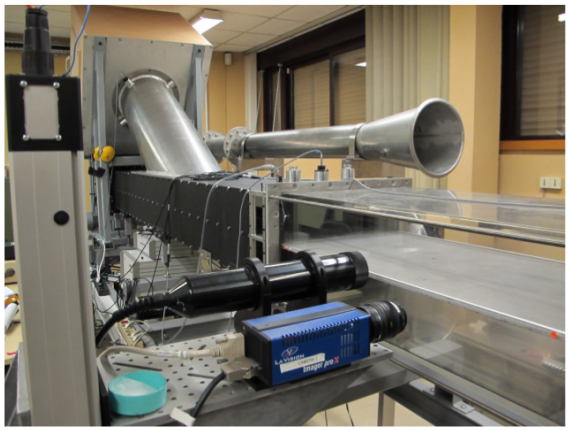


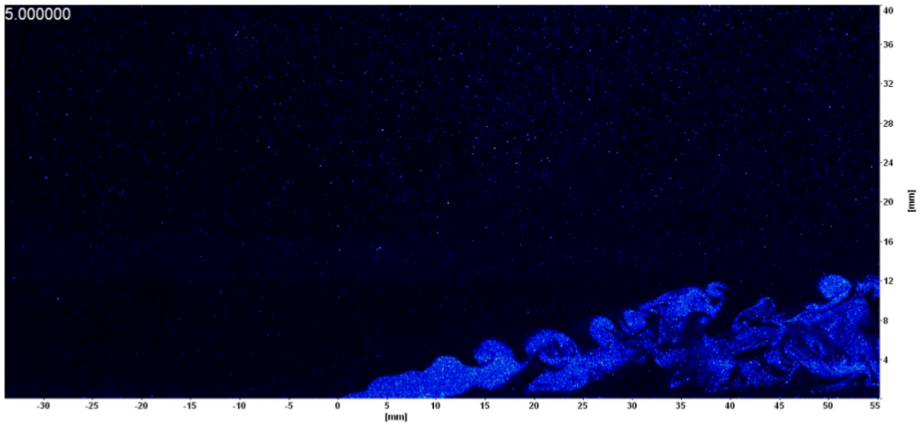


CAGIRE

Simulation and experimentation

- Simulation: `AeroSol` platform.
- Experimentation: MAVERIC test bench.





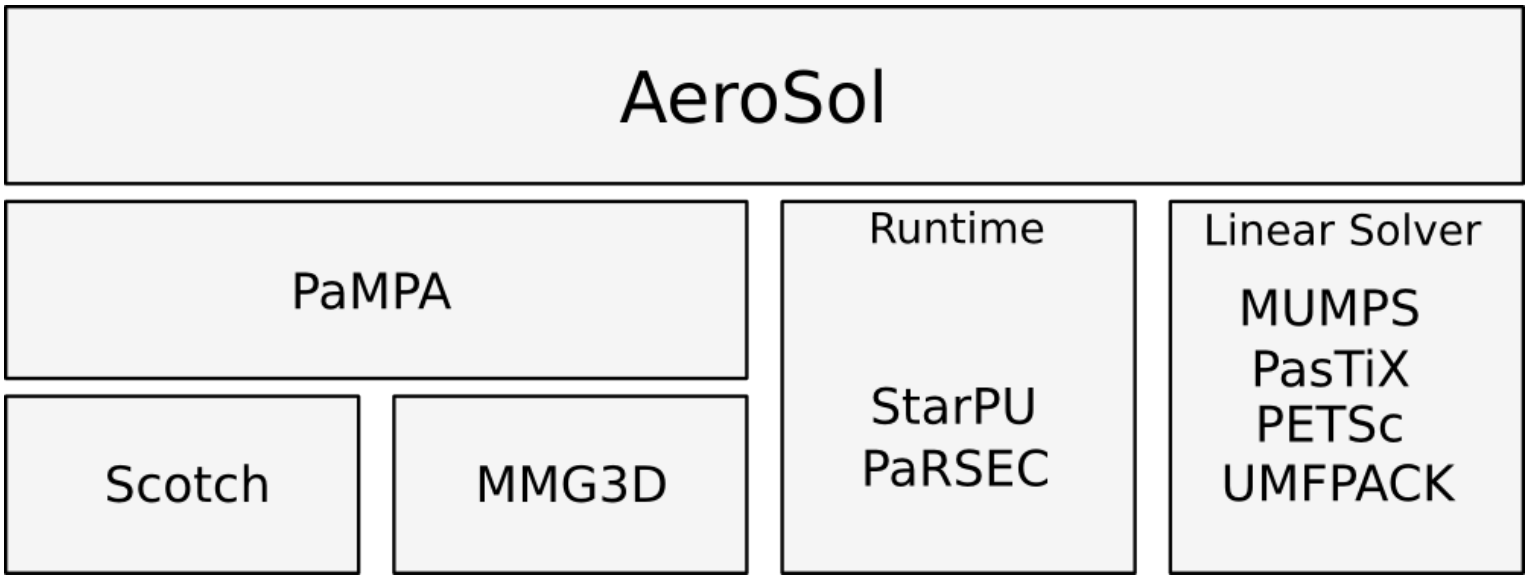
AeroSol

Main goals for the AeroSol platform:

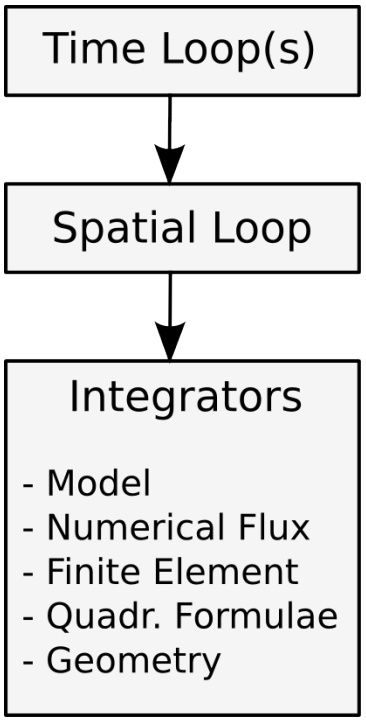
- Collaborative: Anyone can ‘easily‘ contribute.
- Genericity: No restrictions !
- Maintainability: Well designed architecture.
- Performance: Just performance.

2
Software Architecture

Outside AeroSol



Inside AeroSol



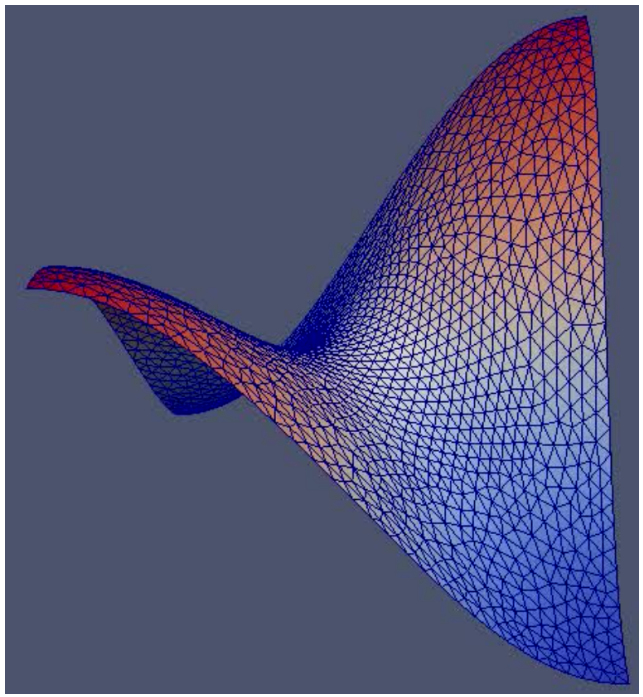
- do operations on abstract data

- compute Residu
- fill Mass Matrix
- solve Ax = b
- iterate over the mesh
- build a local cell/face

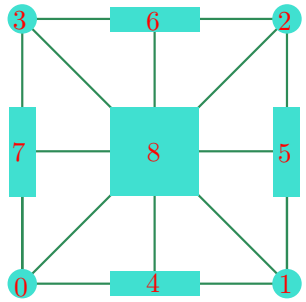
- compute local Jacobian
- compute local Residu

Import - Export

- Import: GMSH in sequential or in parallel.
- Export: Write solution in VTK, XDMF(HDF5), or TecPlot.



Construction of a rich graph. Each element is decomposed hierarchically in a set of entities:



- An entity for the element.
- An entity by face.
- An entity by edge.
- An entity by vertex.

Add some relations between entities:

- Ownership between element and face/edge.
- Rotation between element and face.

$\Rightarrow$ `PaMPA`

Time scheme

- does multi-step, multi-stage;
- does non linear iterations (Newton);
- does abstract operations on vectors;
- commands the spatial scheme for computation.

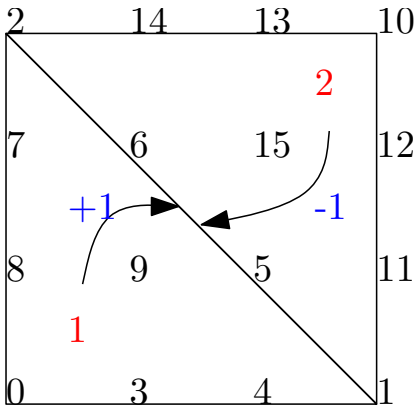
Spatial scheme

- knows the mesh, and iterates over it;
- reconstructs a local cell

  $\rightarrow$ **works for Continuous / Discontinuous**;
- uses `Integrators` to computes quantities;

  $\rightarrow$ **generic for an element**;
  $\rightarrow$ **parametrized by numerical flux in D.G.**;
- assembles a matrix.

Cell reconstruction



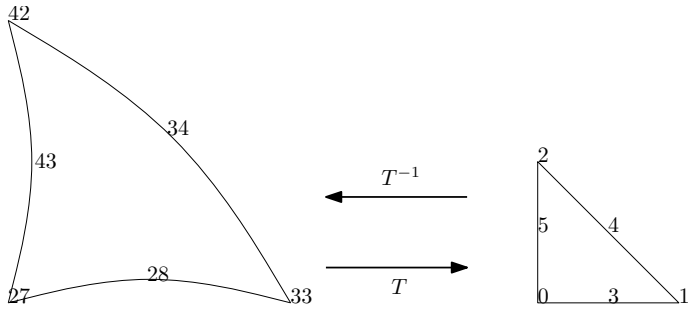
Triangle 1

0 1 2 3 4 5 6 7 8 9

Triangle 2

1 10 2 11 12 13 14 6 5 15

Integrators



- works on reference element;
- uses finite elements;
- uses quadrature formulae;
- uses geometry;
- parametrized by `Model` or `Numerical Flux`;
- computes quantities over an element, or a face.

$\rightarrow$ **works for both continuous and discontinuous**

Matrix class

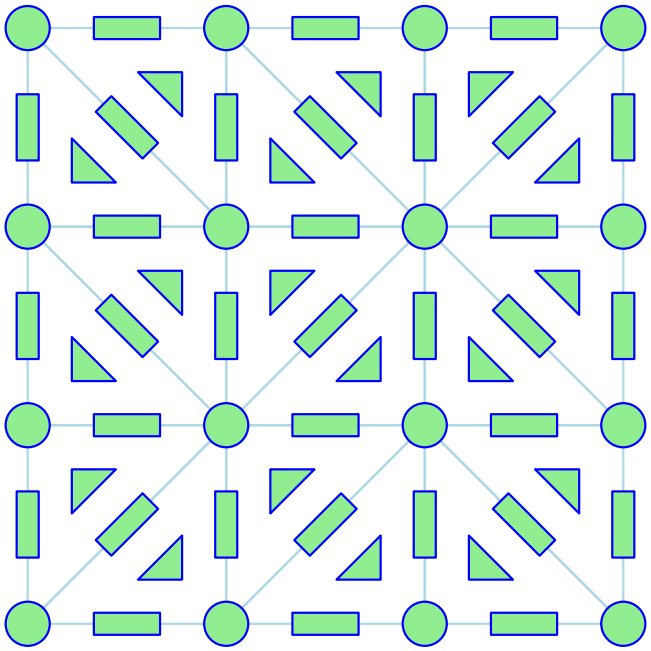
- Distributed matrix.
- Assembled or not.
- Interfaced with many linear solvers:
  - `MUMPS` , `PETSc` , `UMFPACK`

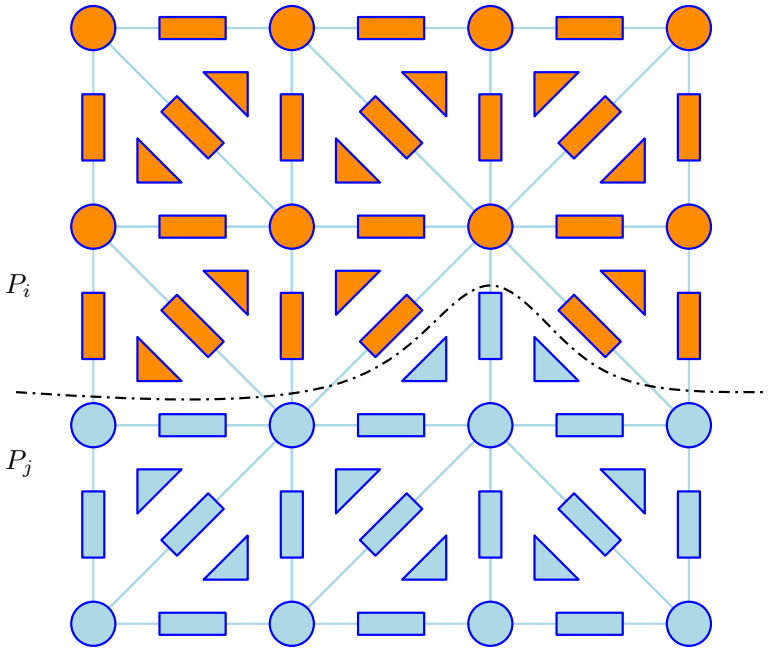
3 Distributed memory level

PaMPA

- handles the mesh and gives visitors;
- redistributes the entities;
- remeshes;
- computes the overlap and do the communications.

Domain Decomposition

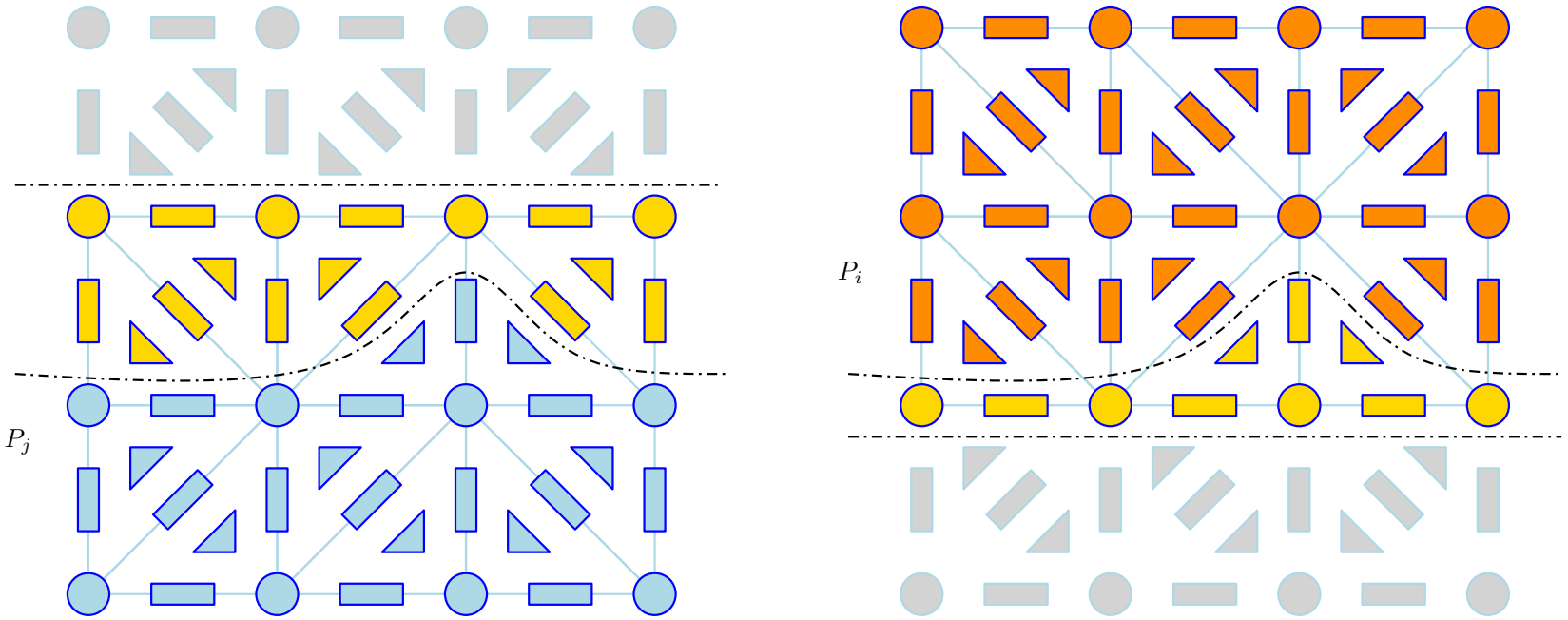




$\rightarrow$ **entites are scattered**

$\rightarrow$ **incomplete elements**

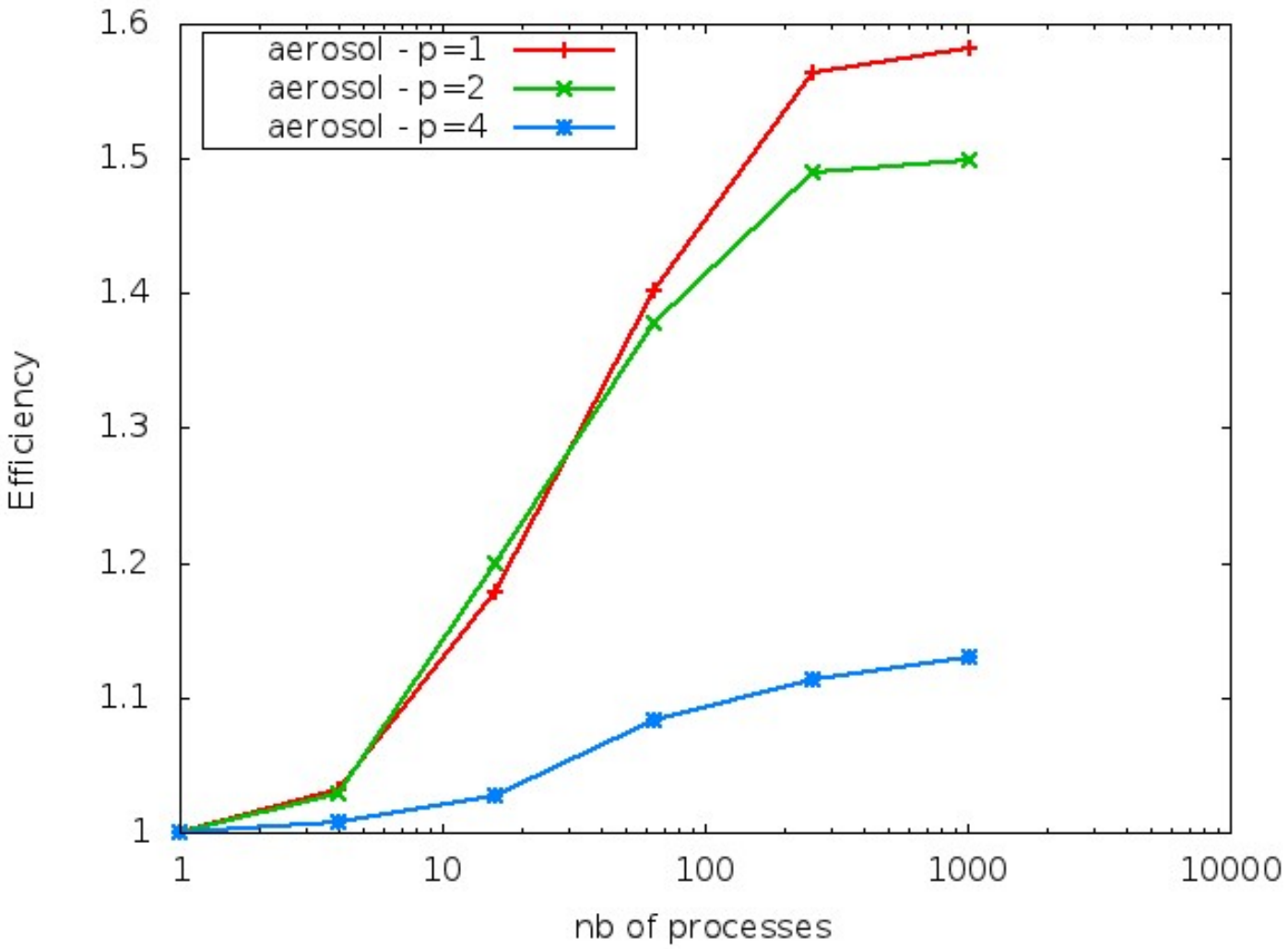
Overlap construction



→ **the overlap completes each incomplete element.**

→ **the overlap gathers entities needed for computations**

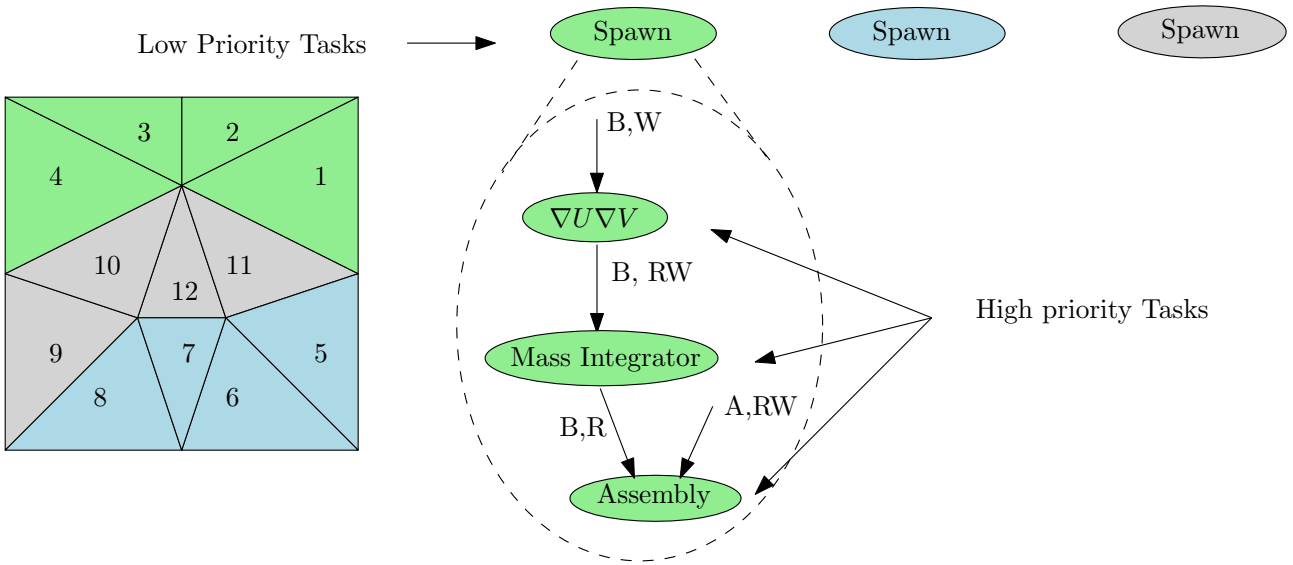
Results on Avakas cluster



4
Shared memory level

Macroelements

We built a set of macroelements.



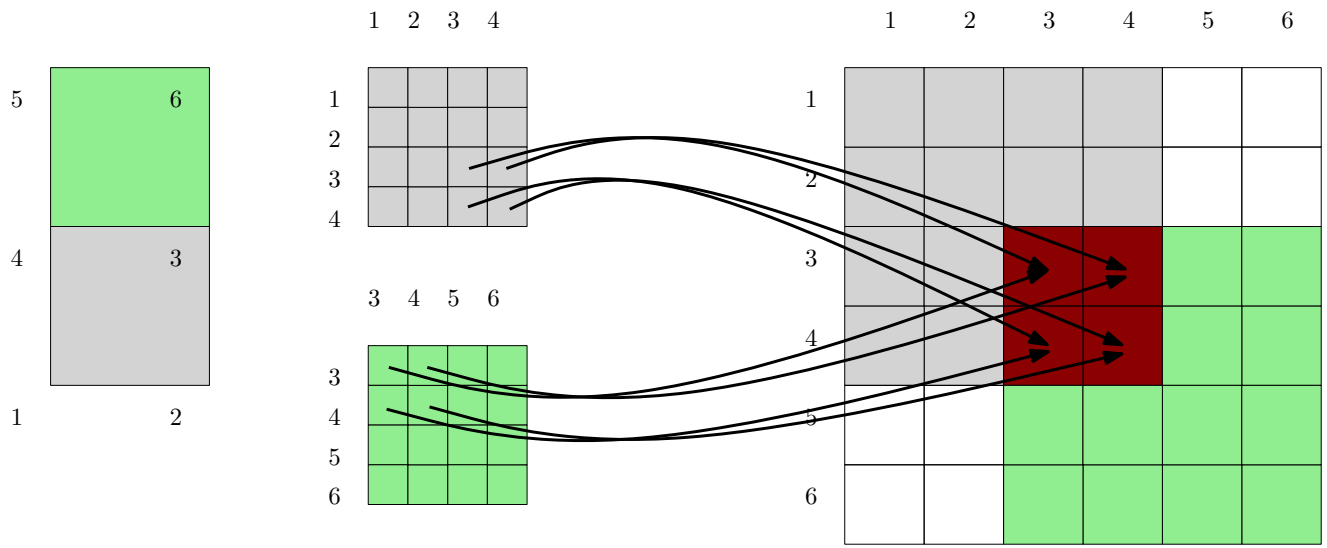
→ **limits the memory consumption;**

→ **limits the number of tasks in the DAG.**

Assembly

We focus on the assembly operations.

Race conditions occur when 2 or more elements share unknowns.

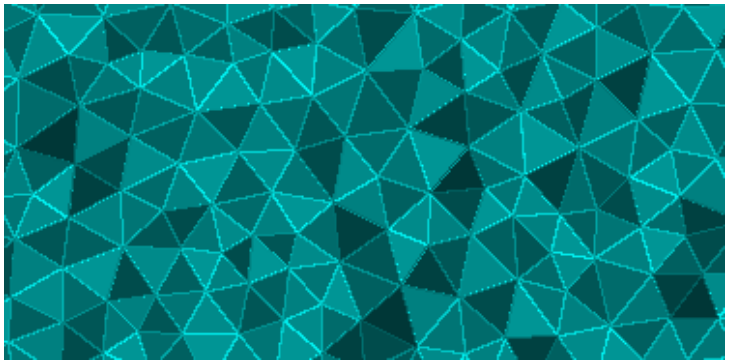


→ **many strategies to do the assembly.**

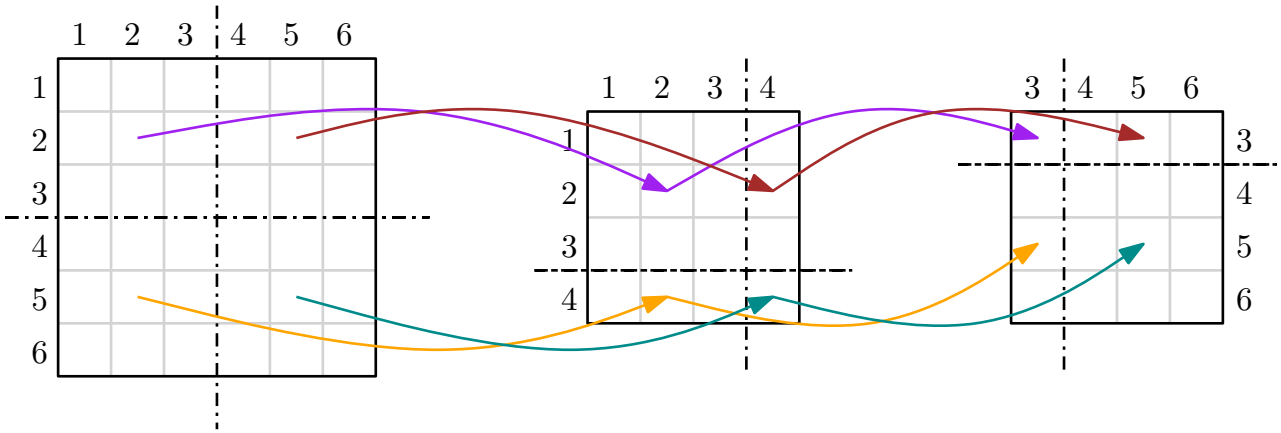
State of the art

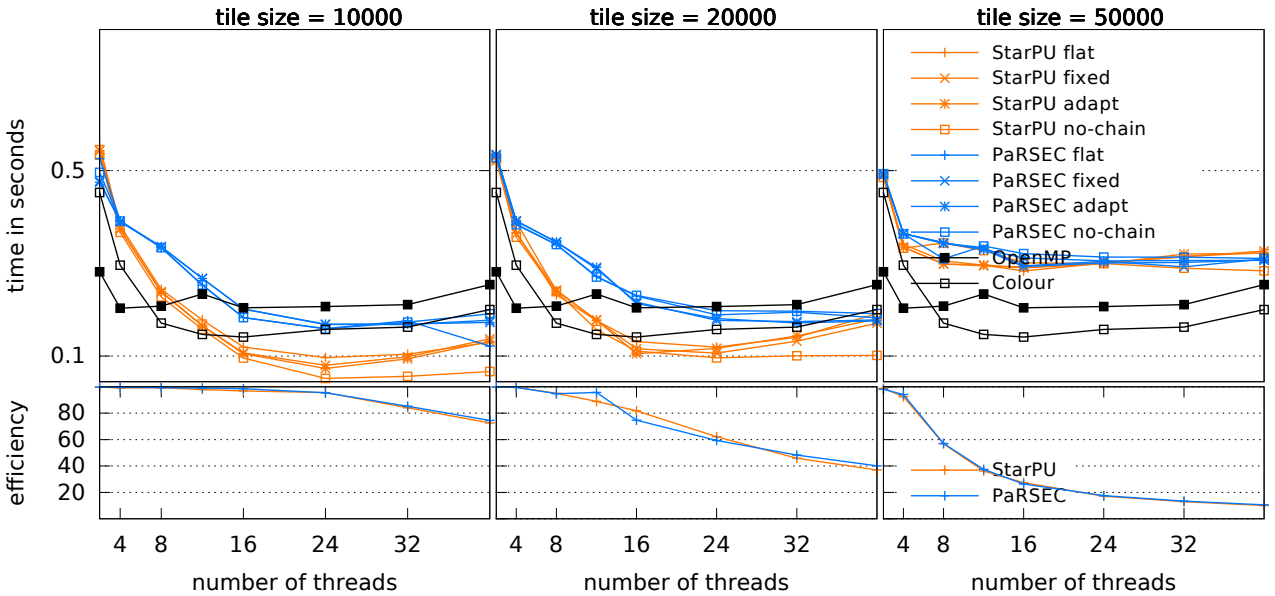
OpenMP: Parallelize the inner loops of the assembly, and treats the element sequentially

Coloration + OpenMP: 2 elements sharing unknowns have different colours.

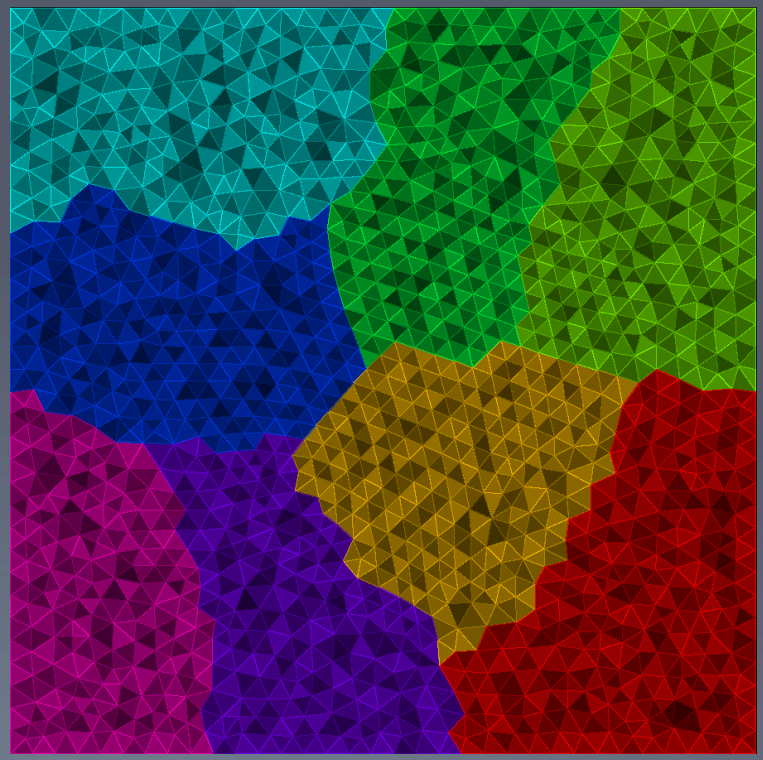


Our strategies





5 Conclusion, perspectives...



Conclusion

- Generic platform with continuous testing.
- Time schemes: Explicit Runge Kutta, Implicit BDF.
- Continuous: CG, Taylor Galerkin, SUPG scheme.
- Discontinuous: Discontinuous Galerkin.
- Model: Advection, advection-diffusion, Euler, N-S (DG).
- FE: Up to 4th order, Dubiner, Legendre, Lagrange.
- Runtimes: Laplacian solved on GPU + CPU with `StarPU` .
- Portability: Avakas, Plafrim clusters. Turing cluster. Compilers : gcc, icc, xlc, Clang.

Perspectives

- DG: MARSU ADT, multigrid methods.
- Scheme: platform continuously improved by ongoing research.
- Model: RANS, LES, Combustion, incompressible free surface flows.
- HPC: efficiency nonlinear iterations in long time integration (interaction non-linear-linear solver), tighter coupling with runtime systems.

Thank you !

INRIA

Bordeaux Sud-Ouest