ParCFD'2020
$32^{nd}$ International Conference on Parallel Computational Fluid Dynamics
May-11-13 2020, Nice FRANCE

# PARALLEL MESH MODIFICATION, LOCATION AND EXTRACTION TARGETING PRE-EXASCALE IN THE CODE_SATURNE CFD TOOL

**Y. FOURNIER[*], J. BONELLE[*] AND C. MOULINEC[†]**

[*] EDF Lab Chatou
Fluid Mechanics, Energy and Environment Department
78400 Chatou, France
e-mail: yvan.fournier@edf.fr, jerome.bonelle@edf.fr

[†]STFC Daresbury Laboratory
Warrington, WA4 4AD, United Kingdom
e-mail: charles.moulinec@stfc.ac.uk

**Key words:** CFD methodology, Parallel methodology, unstructured, in-situ

**Abstract.** This document describes parallel operations involving changes to the mesh structure, parallel identification, and in-situ extraction used in EDF's code_saturne CFD tool, to prepare it for exascale.

## 1 INTRODUCTION

EDF's code_saturne software is a general-purpose Navier-Stokes solver, which has been under development since 1997 [1]. Based mainly on a co-located finite volume method using unstructured polyhedral meshes, it now also includes a large set of Compatible Discrete Operator (CDO) [5] features. Its parallelization paradigm is based on classical spatial partitioning using MPI for communication and a ghost-cell approach to aggregate communication stages. A second-level of parallelism using OpenMP is also available.

The code_saturne tool has been available under the General Public Licence V2 since 2007 [2]. It has also been designated one of the 2 CFD benchmark codes in the European-wide PRACE initiative [3] and run on several of the available tier-0 petascale computing architectures provided in the project.

As an industrial code also targeting extremely large configurations such as Large Eddy Simulations of full nuclear reactors, it is essential for the toolchain to be straightforward. This implies that operations such as domain partitioning, generating output, or restarting a calculation on a different number of processes should be as transparent to the user as possible. When these operations can be done "in situ", we get the added benefit of avoiding expensive IO, and removing complex dependencies between compute stages having different resource requirements.

## 2 BASE TOOLCHAIN AND ARCHITECTURE

To enable running on a wide range of machines, the code_saturne toolchain naturally separates tools dedicated to lightweight and especially interactive operations, such as the GUI, and the main solver itself, which should be able to run at a large scale in the batch environments of supercomputers.

### 2.1 USE OF THIRD-PARTY LIBRARIES

Strong dependencies on third-party libraries inside the main solver have been avoided as much as possible to ensure portability. Third-party libraries can extend features, but are all optional, so porting to newer architectures is much easier, and can be done in multiple steps (starting with a "bare bones" install, then adding bells and whistles).

Optional libraries include PT-SCOTCH and ParMETIS (in addition to native Space-Filling Curve partitioners), the PETSc library (in addition to native linear algebra features), the ParaView/Catalyst library for in-situ postprocessing, MED and CGNS to support the associated mesh formats, and fluid property libraries.

## 3 PARALLEL IO AND PARTITIONING

A key tenet of code_saturne solver developement is that mesh-based data is distributed at all stages, and no single MPI process should ever need to contain a single global array. This implies that data must be read in parallel, or at worse read by chunks and distributed immediately i.e. "funnelled". For this, direct MPI-IO is mainly used. Partitioning algorithms must of course also be distributed.

Distribution of data at such an early stage results in an initial distributon which may be far from optimal, and some mesh entities and their adjacent entities (such as faces and vertices) may not even be read on the same core. To handle this, we use redistribution operators for different adjacency configurations based on global ids. Since these operators use all to all algorithms internally, which been encapsulized in a high-level API, allowing instrumentation and more importantly choice of different algorithms, to better adapt to very high MPI rank counts.

## 4 PARALLEL MESH MODIFICATION

The support of polyhedral elements in code_saturne has been both very useful and constraining, as many third-party tools or libraries do not support those elements well, if at all. This support is the basis of a distributed algorithm for non-conforming mesh joining, described in older publications [4], and has allowed to work around many serial meshing tool's limitations by assembling piecewise meshes on-the-fly.

In recent years, features allowing to reduce the time for mesh generation have been added, all of which are implemented in parallel:

- inserting boundaries on selected interior face sets, possibly separating the mesh into multiple zones; vertices are duplicated where necessary, and complex (non-manifold) surfaces can be handled;

- extruding selected boundary faces along chosen directions; the number and thickness of the extrusion layers may be defined to vary locally;
- inserting viscous layers combining the above feature with a mesh deformation algorithm. This can be done in the setup stage, or at the end of a computation, where flow field data can be used to adapt the thickness for a subsequent run;
- adding a local mesh refinement using predefined templates for regular elements, and a generic one for polyhedra.

Work has also started on making the mesh refinement locally reversible, to allow for full mesh adaptation features.

## 5 PARALLEL MESH LOCATION

Mapping from one mesh to another is increasingly used, and is involved whether restarting from a different mesh, mapping internally from one boundary to another, coupling with another code, or in algorithms such as mesh joining.

Multiple algorithms are used, with different performance properties and APIs. The internal APIs used are quite high-level, and lower level optimizations improving the scalability should have a minimal maintenance impact. This will be detailed in another publication.

## 6 PARALLEL MESH AND DATA EXTRACTION

With computations involving meshes with hundred of billions of cells, the volume of data produced can be huge, especially in the context of transient flows.

In addition to the performance impact of file IO, in many cases, “post-hoc” visualization or processing tools may be unable to handle huge data sets, whether due to tool scalability, file format limitations, or postprocessing resources availability. And for all this, the amount of data that is finally used is often much smaller than the data produced.

From the onset, code_saturne has allowed users to define additional operations on the data, which is often used to compute balances, statistics, and extract values in specific regions of interest. Extracting a subset instead of a full computational mesh for postprocessing has been almost systematically used on our larger computations so as to limit postprocessing to “manageable” quantities.

In recent years, we have made an increasing use of ParaView/Catalyst [6], so as to offer a more complete extraction capability, including most of the visualization capabilites available to smaller cases, albeit in an in-situ manner. Though this may require some additional preparation, it allows handling complex postprocessing operations and generating advanced visualizations with a very small I/O footprint. As the fields of in-situ and in-transit processing is evolving fast, more options will probably be added in the future.

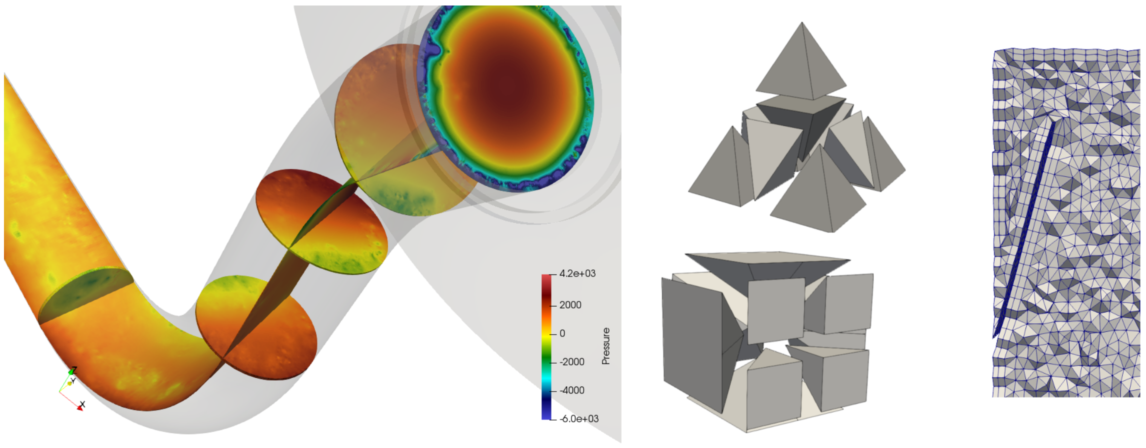

**Figure 1**: left to right: in-situ defined extracts, refinement templates, viscous layer insertion

## 7 CONCLUSIONS

Although code_saturne may be considered as a "legacy" code, enforcing some precautions over its developement and progressively migrating an ever-increasing portion of its toolchain to built-in or library-based in-situ operations using distributed and mostly scalable algorithms, it has maintained the ability to run smoothly on tier-1 and tier-0 machines.

## REFERENCES

[1] F. Archambeau, N. Mechitoua and M. Sakiz "*Code_Saturne*: a finite volume code for the computation of turbulent incompressible flows  industrial applications", Int. J. on Finite Volumes, 2004.

[2] https://code-saturne.org

[3] https://www.prace-ri.eu

[4] Y. Fournier, J. Bonelle, P. Vezolle, C. Moulinec, A.G. Sunderland, "An Automatic Mesh Joining Approach for CFD to Reach Billion Cell Simulations", PARENG2011, April 2011, Ajaccio, Corsica, France

[5] J. Bonelle, A. Ern, "Analysis of Compatible Discrete Operator schemes for elliptic problems on polyhedral meshes", Multiscale problems and techniques, ESAIM: M2AN 48 (2014) 553581, DOIhttp://dx.doi.org/10.1051/m2an/2013104

[6] A. Ribes, B. Lorendeau, J. Jomier, Y. Fournier, "In-Situ Visualization in Computational Fluid Dynamics Using Open-Source tools: Integration of Catalyst into Code_Saturne", Topological and Statistical Methods for Complex Data – Tackling Large-Scale, High-Dimensional, and Multivariate Data Sets, Springer. Pages 21-37 (2015) ISBN 978-3-662-44899-1.