CSMA 2024
16ème Colloque National en Calcul des Structures
13-17 Mai 2024, Presqu'île de Giens (Var)

salome_meca : A mechanical simulation platform tailored for studies and research

A. Gangnant[1], N. Tardieu[1], M. Abbas[1], G. Ferté[1]

[1] *EDF R&D ,Département Electrotechnique et Mécanique des Structures, {alexandre.gangnant,nicolas.tardieu,mickael.abbas,guilhem.ferte}@edf.fr*

Abstract — salome_meca is an open source, dedicated platform designed for mechanical simulations. Its primary objective is to provide a comprehensive software environment that enables the entire numerical simulation process, encompassing CAD, meshing, computation, post-processing, result visualisation,and even parametric studies. The platform embedded the code_aster solver developped by EDF R&D . Over the past few years, code_aster has undergone further development with a focus on high-performance computing (HPC) to address intricate and high degrees of freedom (DoF) structures.

Mots clés — code_aster , salome, salome_meca , numerical simulation, open source, HPC

1 The *code_aster* solver

Since 2001, code_aster has been available as open-source software under the GNU General Public License. The software, its source code, test cases, and documentation are freely accessible through open access [1]. Over the course of 34 years, code_aster has achieved global recognition and has brought together users from around the world through its website forum, making it an impressive success in terms of distribution and community involvement.

1.1 Presentation of *code_aster*

Safety and availability of mechanical installations and civil engineering structures at EDF necessitate justification for operational, repair, and replacement purposes.

These justifications primarily rely on non-linear mechanical modeling. Therefore, in contrast to the standard functionalities of conventional finite element mechanical software, code_aster distinguishes itself by consolidating various scientific research efforts addressing these safety challenges.

Ultimately, the code_aster solver ensures the effective management of this scientific research and its seamless integration into engineering units.

The ambition for code_aster is twofold :

- To provide an updated, powerful simulation software that is stable and robust for expert-level studies, all within a development and dissemination framework maintained under rigorous quality assurance.

- To facilitate the integration and consolidation of numerical mechanical models from EDF R&D .

Linked to the two mentionned objectives, developping its own code ensures the capitalisation of the R&D and and facilitates fast knowledge transfer to the engineering team. Such an endeavor may become quite intricate when dealing with commercial codes. As EDF operates rather than manufactures, this R&D effort is entirely unique. EDF has to, in fact, substantiate the lifetime of its equipment and infrastructure, both from economic and regulatory perspectives.

The primary focus code_aster is to consistently enhance the quality and usability of the code by expanding its usage. This, in turn, positions us as a key contributor to partnerships in the field of simulation.

1.2 Capabilities of *code_aster*

code_aster incorporates dinstinct numerical models for :

- Simulating the ageing of materials and structures: creep, fatigue, damage, fracture mechanics, porous materials, and more.
- Addressing specific challenges related to nuclear operations: Soil-Structure interaction, seismic analysis, regulatory computations, and fuel assembly modeling, among others.

1.3 *code_aster* as a Python module

Since version 15.4 (the current one being 16.4), code_aster has transformed into a straightforward and traditional Python module. This development marks a significant improvement in the interaction between the solver and the user. Without the need for memory duplication, all lower-level objects (Fortran) are now directly accessible from the Python environment. code_aster is now ready for the next 30 years.

```
1
2     import code_aster
3     import numpy as np
4     code_aster.init("--test")
5     test=code_aster.TestCase()
6
7     # Creation of the mesh
8     mesh = code_aster.Mesh()
9     mesh.readMedFile("zzzz255a.mmed")
10
11    # Creation of the model
12    model = code_aster.Model(mesh)
13    model.addModelingOnGroupOfCells(code_aster.Physics.Mechanics,
14                                    code_aster.Modelings.Tridimensional,"ALL")
15    model.build()
16
17
```

This new mode of interaction complements the "historic" mode. Consequently, the command file can be employed in a hybrid manner, combining the use of Python objects and traditional code_aster commands. The integration of Python into code_aster facilitates the initiation of high-performance computing (HPC) capabilities.

1.4 *code_aster* improving the HPC

High-performance computing (HPC) is an ongoing area of extensive development. Substantial work has been carried out in both the meshing and solver aspects.

Through the partitioning of meshes and the implementation of "ghost" elements technology at interfaces, the HPC mode enables the computation of large-scale models with heavy computa-

tional demands, reaching hundreds of millions of degrees of freedom (DoF), all while minimizing memory usage. Here are some key HPC functionalities:

- A new parallel MED partitioner, `PTSCOTCH`, can be utilised during mesh reading (requires the activation of new subdomain parallelism);
- Parallel I/O capabilities, allowing data storage in a single MED file;
- Fully HPC-compatible linear mechanical and thermal solvers, such as `MECA_STATIQUE` and `THER_LINEAIRE`;
- Introduction of a new preconditioner `HPDDM` designed for handling large linear systems (`SOLVEUR=_F(METHODE="PETSC", PRE_COND="HPDDM")`).

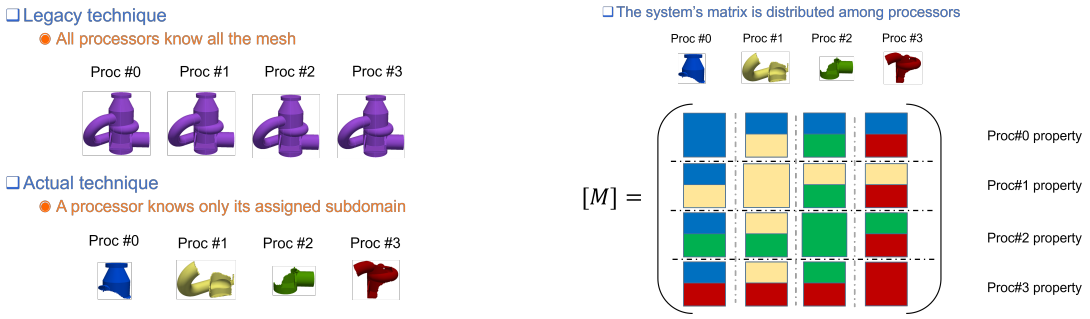

Figure 1: HPC in code_aster (mesh partioner : subdomain approach)

As an exemple, the Figure 2 below displays the gain of the use of HPC on a simple academic problem such as a ultra-refined cube.

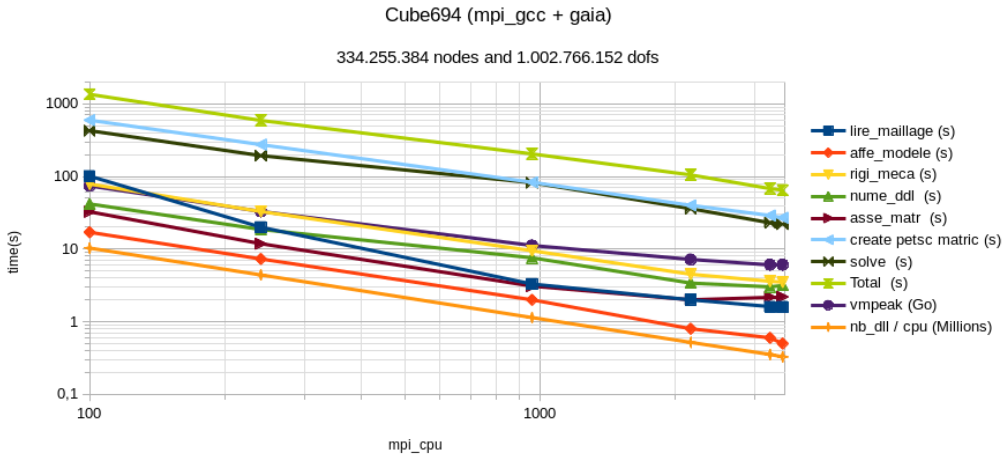

Figure 2: Example of computation time benifits through the use of HPC in code_aster

Work is currently underway to incorporate HPC capabilities into the Nonlinear solver `MECA_NON_LINE`, introducing new possibilities like Scalable Nonlinear Equation Solvers (SNES). These computational codes are frequently of strategic importance and facilitate significant technological advancements in the understanding of physical phenomena.

2 The *salome_meca* platform

salome_meca is a oriented mechanical dedicated platform builds on *Salome* basis.

2.1 The *Salome* platform : the kernel

For several years, EDF R&D has been actively developing various computational codes within the simulation framework (code_aster , Code_Saturne , Syrthes, and others).

Salome provides a comprehensive and unified user-interaction environment for pre- and post-processing, which is essential for utilizing the mentioned codes. Consequently, this platform streamlines the creation of data models while minimising the onboarding costs for new users. Additionally, its adaptable architecture enhances interoperability between CAD, solvers, and the implementation of coupling between computational codes within a diverse distributed environment, which is particularly advantageous for multi-physics coupling computations.

The *Salome* platform is developed as a consortium effort and is released as OpenSource under the LGPL license ([2]). Working with an Open Source code enables computational code developers to primarily concentrate on their core expertise, such as physics and numerical methods, while also enhancing the capitalisation of resources.

2.2 *salome_meca* platform : dedicated skills tools and modules

The ambition of *salome_meca* is to offer the complete simulation chain, including CAD, meshing, mechanical computation, post-processing, and results visualisation.

The salome_meca 2023 platform (released at the end of 2023) enable to gather in an unique environment:

1. **CAD** modules (`GEOM` and `SHAPER`) ;
2. **Mesh**module (`SMESH`) which includes several 3D, 2D, and 1D mesh algorithms ;
3. ***code_aster*** (1) as the implicit mechanical solver, with the latest stable version being 16.4. ;
4. **AsterStudy**, the latest GUI generation for *code_aster* ;
5. **Persalys**, a GUI dedicated to handling uncertainty and variability management (with AsterStudy integration) [3] ;
6. **ADAO**, designed to assist with data assimilation or optimization methodologies, compatible with other modules or simulation codes, and usable within both Python and SALOME contexts ;
7. **Paravis**, a visualisation and post-processing module based on Paraview [4].

Note that one strength of the SALOME Platform lies in its ability to facilitate the easy implementation of specialised tools and modules tailored to specific applications, such as salome_meca . For instance, within EDF, on a non-distributed version (EDF Only), salome_meca incorporates dedicated EDF modules for test-analysis correlation, as well as modules for civil engineering, piping, rotating machine simulations, and more.

3 Références

References

[1] *Official code_aster website* : www.code-aster.org

[2] *Official Salome website* : www.salome-platform.org

[3] *Official Persalys website* : https://www.persalys.fr

[4] *Official Paraview website* : www.paraview.org