Towards massive parallelism
Massive parallelism: the genesis

- Fundamental changes in the mind and functioning of code_aster

- From a simple question
  - Modification of machines and uses
  - Is it possible to transform code_aster into HPC code?

- Answer

YES!!
Massive parallelism: the solution

- **Initial problem: memory consumption**

- **Domain Cutting**
  - code_aster parallelism today

  ![Parallelism Example](image)

  - code_aster HPC

  ![HPC Example](image)

  - => Reconstruction of a global problem
Massive parallelism: the results (1/3)

- First results, demonstration of interest
  - Academic test comparison with code_aster “legacy”

![Graph showing performance comparison between two codes for a cube elastic test with 700,000 degrees of freedom on 16 processors.](image-url)
Massive parallelism: the results (2/3)

- More ambitious

<table>
<thead>
<tr>
<th>Number of Dof</th>
<th>Number of procs</th>
<th>Solving time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128 625 000</td>
<td>360</td>
<td>1477</td>
</tr>
<tr>
<td>128 625 000</td>
<td>720</td>
<td>820</td>
</tr>
<tr>
<td>128 625 000</td>
<td>1080</td>
<td>627</td>
</tr>
</tbody>
</table>

87% of parallel efficiency between 720 and 1080 processors

- Huge reduction of memory consumption and simulation times

- Enlarges the range of possibilities in terms of model size
Massive parallelism: the results (3/3)

- Elasto-plastic computation (static and non-linear)

<table>
<thead>
<tr>
<th>Dof</th>
<th>Procs</th>
<th>Solving time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 547 411</td>
<td>1 080</td>
<td>56</td>
</tr>
<tr>
<td>41 051 928</td>
<td>1 080</td>
<td>257</td>
</tr>
</tbody>
</table>

  - code_aster: 3 h 30 min
  - code_aster HPC: 4 min

- Sedimentary basin calculation (linear dynamic)

<table>
<thead>
<tr>
<th>Dof</th>
<th>Procs</th>
<th>Solving time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25 842 015</td>
<td>270</td>
<td>1.5</td>
</tr>
<tr>
<td>185 911 227</td>
<td>1 080</td>
<td>4.8</td>
</tr>
</tbody>
</table>

  - Extrapolation for 2000 time steps: 3 h
Massive parallelism: the integration

- **Difficulty:** each processor could think it is alone
  - Data handling risk

- **Solution:** add encapsulation
  - Object oriented programming
  - Data access control

- **Fundamental change in architecture of code_aster**
  - Adding of a new programming language: **C++**
  - Replacement of the “aster supervisor” by Python supervision
Massive parallelism: the architecture

- Imperative: preserve the existing
  - 29 years of Fortran
  - 29 years of user habits (Aster syntax)

```python
Python shell: >>> import code_aster
```

- What doesn’t change
- What’s new

- Python interface to data structures
  - C++ wrapping
  - Boost-Python

- Python “Macro-commands”
- Fortran operators (OP****)
- Aster commands (LIRE_MAILLAGE, ...)
- Data structures (sd_maillage, ...)
- Python interface to data structures
Massive parallelism: the collaborative project (1/2)

- Subsidized collaborative project
  - EDF R&D
  - Public research institution: IFPEN
  - Academic partner: LMT Cachan, CERFACS
  - SME:
    - V&V: NECS
    - Interoperability and deployment: PhiMeca

- Shared target:
  
  Develop and make available a more powerful and interoperable HPC version of code_aster
Massive parallelism: the collaborative project (2/2)

- **PAMSIM (PA rallélisme Massif en SI mulation numérique pour la Mécanique)**

- A 3-year project ending in late 2018

- Late 2018
  - HPC version of code_aster
  - With same V&V as code_aster “legacy”
  - Deployed on external supercomputers
Massive parallelism: the consequences (1/2)

- What changes for users?
  - In practice
    ```
    my_file.comm:
    DEBUT();
    # (...)
    FIN();
    ```
    ```
    my_file.py:
    import code_aster
    code_aster.init()
    # (...)
    ```

- No other changes to the command file
  - The most transparent solution possible for the user

- “Old-fashion” parallelism continues to exist
  - The two modes co-exist
Massive parallelism: the consequences (2/2)

- What changes for developers?

- New architecture designed to preserve habits
  - Ease of change management

- In the short term: nothing changes abruptly

- Progressively:
  - The core team will develop in C++
  - Occasional developers will work in Python
Massive parallelism: the new things

- **The HPC, of course**
  - Linear and non-linear mechanical calculations
  - Always with the same license cost

- **New access to data structures (python shell)**
  - To be enriched according to needs
    ```python
    # (...) 
    mesh = LIRE_MAILLAGE()
    
    # get coordinates of the mesh
    coord = mesh.getCoordinates()
    
    # print coordinates
    print "coord[3]", coord[3]
    ```

Access to value without copy
Massive parallelism: the outlooks

- Developing and using AsterHPC's capabilities
  - Valorisation through use on critical studies for EDF
    - Site effect for the seismic justification of our nuclear power plants
      - 200 M of physical ddl in linear dynamics over 4 000 time steps
      - Contribution to the robustness of our seismic safety demonstrations
    - Sizing of PWR nuclear waste repository galleries
      - 50 M of ddl in very non-linear quasi-static modelling
      - Costs avoided in billions €

- Continue to enrich the interfaces of the objects
  - Facilitate data access and interfacing

- Replacement of “official” code_aster by the new version
  - Replacement in 2019
Merci de votre attention !