A Containerized CAD to FEM Infrastructure Solution Based on Open Source Projects

Olga Sedova[1]* and Oleg Iakushkin[1]

[1] *Department of Computational Methods in Continuum Mechanics, Saint Petersburg State University*

Universitetskaya nab. 7/9, 199034, St. Petersburg, Russia
e-mail: o.s.sedova@spbu.ru

Abstract

In this paper, we consider concept of a system, encapsulating different steps from the creation of model geometry via CAD to various tests and stress testing simulations using FEM. This concept considers an engineer that can act as a programmer, and that all necessary steps from modelling to the declaration of boundary conditions are provided in form of source code. We propose an implementation of such software. In its development, we are using: Python programming language; Jupyter framework for visual representation of user's workspace; pythonOCC and FeniCS libraries that implement CAD and FEM algorithms; utilities such as GMSH and VTK.
Our platform prototype is tested on a dynamically expandable multitenant cloud system and is presented to end users in form of a web service, providing all the necessary computing resources to users on demand. At the same time system can be deployed locally for prototyping and non-resource-intensive workloads. To achieve this, the system was isolated via Docker containerization engine.

Keywords: FEM, CAD, Cloud, Software, opensource, programming

1. Introduction

At the prototyping stage tasks of CAD geometry modelling systems often appears in conjunction with structural analysis via FEM. Yet the variety of file formats and application interfaces restrict smooth flow that reflects on the rate of development progress and slows down speed of iterations. This leads us to the task of creating a system that would bring together the work process from CAD to FEM within a single application. However, in practice, the solution of such problem encounters three main difficulties: providing the necessary functionality within a single user interface, computing needs of various task segments, the choice of parameters to be displayed to a user at any given moment. [1]

2. Our Concept

We propose to consider an engineering as a programmer. In the case of pre-built components that implement the processing logic of all the necessary tasks for CAD and FEM logic, an engineer will need to describe all the necessary modelling and boundary conditions in form of the source code at a high level of abstraction.

The proposed approach shows how to make a computer program, but for rapid prototyping we need the ability of visual results presentation at various stages across model construction and testing stages. In other words, user must be able to see the changes at any step of the code, enter the previously saved position or edit a set of options and then rerun the program from a given point. Such opportunity is provided by interactive shell programming languages. Maple and SageMath are good examples of such platforms.

However, programmers need building blocks of programs - libraries for CAD and FEM operations. Some of them may require specific and different physical infrastructures to achieve maximum performance. A range from coprocessors and many-core systems to clusters of interconnected nodes. Technological stack plays its role as compatibility of CUDA, MPI, OpenMP programming interfaces may vary [2, 3]. At the same time, it is important for the end user convenience, for the library to be integrated into a visual shell environment to enable fast interactive experience.

3. Presented Implementation

We have developed a prototype implementation of the proposed concept, here are its main components:

- Python was chosen as a main system programming language – for system developers and end users;
- Jupyter framework was used as an interactive shell that delivers a single visual experience for all the user needs;
- pythonOCC library provided a set of CAD tools and a three-dimensional models viewer in a form consumable by a web browser;
- FeniCS library is used as an implementation of FEM algorithms. It supports direct integration with Jupyter visual environment;
- Utilities GMSH and VTK were used at intermediate stages for the conversion of model formats and visualization;
- Tuned system was containerized for isolation and fast deployment.

Selected libraries allow us to use possibilities GPU co-processors via OpenCL or CUDA, while MPI allows us to distribute computation across cluster nodes.

Platform for is deployed on a dynamically expanding cloud cluster in a containerized form. Service is provided in the form of a web-service, accessible from any web browser. End user code execution is occurring on demand, it allows a flexible dynamic scaling of our virtual cluster [4, 5]. For multiple users to access our platform system we use JupyterHub.

* Corresponding author. Tel.: +7 812 428 4492.

4. Conclusion

We present a prototype of a system that demonstrates some aspects of an engineer's work if it would be entirely focused on coding inside of an interactive shell in a web browser with support for visualization of three-dimensional models with rendering on both the client and server side; all stages of prototyping are done using only a single application running on a dynamically allocated resource of a cloud cluster.

5. Acknowledgment

This work was supported by the Russian Foundation for Basic Research (project N 16-07-01113).

References

[1] Iakushkin, O., Kondratiuk A., Sedova O., Grishkin V., Jupyter extension for creating CAD designs and their subsequent analysis by the finite element method, *International Conference on Computational Science and Its Applications*, Vol. 1787, pp. 530-534, 2016.

[2] Iakushkin, O., Shichkina Y., Sedova O., Petri Nets for Modelling of Message Passing Middleware in Cloud Computing Environments, *International Conference on Computational Science and Its Applications*, Vol. 9787, pp. 390-402, 2016.

[3] Iakushkin, O., Grishkin V., Messaging middleware for cloud applications: Extending brokerless approach, *International Conference on Computational Science and Its Applications*, Vol. 9787, pp. 1-4, 2014.

[4] Fatkina, A., Tikhonov N., Iajushkin O., Application of GPGPUs and Multicore CPUs in Optimization of Some of the MpdRoot Codes, *25th Russian Particle Accelerator Conf.(RuPAC'16)*, WEPSB025, pp. 416-418, 2017.

[5] Iakushkin, O., Sedova O., Grishkin V., Application control and horizontal scaling in modern cloud middleware, *International Conference on Transactions on Computational Science XXVII*, Vol. 9570, pp. 81-96, 2016.