SHAPE 10th Call - Application Form

Project Title	RF Solver on multi GPUs
Company Name	OPEN ENGINEERING
Company Address	Allée des Noisetiers 2, 4031 Angleur, BELGIUM
Company Website	https://www.open-engineering.com
Company Number	BE0476335227
Number of Employees	13
Contact Name	De Vincenzo Pascal
Contact Position	General Manager
Contact Email	p.devincenzo@open-engineering.com
Contact Phone	+32499510489

Has the company worked with PRACE before?

No

What does the company do?

The Open Engineering was created in 2001. It is a high-tech SME, located in Liège (Belgium).

Our company is active in the Computer-Aided Engineering (CAE) market. We design, develop, and sell the OOFELIE::Multiphysics software.

OOFELIE::Multiphysics is used to conceptualize, design, analyse, and optimize various types of systems before starting time-consuming and costly build-and-test cycles.

OOFELIE::Multiphysics's features are focused on three major domains where they are used to predict the behaviour of:

- Sensors and actuators, including MEMS (Micro-Electro-Mechanical Systems) and Microsystems.
- Optomechanical systems including MOEMS (Micro-Opto-Electro-Mechanical Systems).
- Multidisciplinary systems where an interaction exists between a fluid medium and a structure (F.S.I. - Fluid Structure Interaction).

These activities encompass a broad range of products in the aeronautics, space, defence, automotive, shipbuilding, and consumer electronics markets.

Together with those software activities, Open Engineering provides customization services of its existing solution to customers' needs, and performs engineering services.

At Open Engineering, we are thus familiar with numerical simulation and software development. Our standard OOFELIE::Multiphysics solver is written in C++. It makes mainly use of the finite element (FEM) and boundary element (BEM) methods combined with implicit resolution strategies. Parts of the code have been parallelized with OpenMP; and some linear solvers are also parallelized with MPI. Internally, we have a small cluster made of Intel workstations attached to an Infiniband switch. We also have expertise on the development of scientific algorithms for GPUs using CUDA. However, we currently lack the expertise on multi-GPU programming.

Project Abstract

Recently, we have developed an electromagnetic wave propagation solver for RF MEMS applications. Its core algorithm is based on the FDTD method.

Two versions have been developed so far respectively with C++ and with CUDA.

The C++ version has been parallelized with OpenMP. It has been tested on single CPU workstations and dual Xeon servers. The CUDA version has been tested on GPGPU and TESLA accelerators (K40 & V100).

From "Roof model", we have gained the knowledge that our CUDA implementation benefits from high memory bandwidth (the higher the better).

However, GPUs are often limited in terms of memory (up to 32GB for TESLA V100) compared to workstations.

Therefore, our goals are (i) to develop a multi GPU version of our CUDA solver to address problems that do not fit in the main memory of a single GPU; and (ii) to analyse its performance on an HPC multi-GPUs host.

Industrial relevance and potential business impact

Our business model is based on the sale of licenses and engineering services. With each new functionality, we start by delivering services (using the code ourselves). Then, after industrialisation of this functionality, we can start selling licences. Today, our RF solver is used for engineering activities.

There are multiple benefits expected from this project:

1. New knowledge in development of parallel solvers on multiple GPU with CUDA thanks to the PRACE partner. This new expertise will then be used for other algorithms available in OOFELIE.

2. The updated version of our RF solver that will permit us to solve problems that do not fit in the main memory of a GPU. For our engineering service activities, it means that we can resolve RF applications with more details and thus more accuracy.

3. The updated version of our RF solver being capable of benefiting from the computing power available when multiple GPUs are attached to a single host. This will allow us to solve larger problems with a reduced computation time. This will also speed up the execution of numerical design of experiments (DoE). Again, this will allow us to deliver, faster, more analysis results to our customers.

4. Insight to prepare a new version of the solver that will combine both MPI and multiple GPUs.

The positive outcome of this project will thus contribute to (i) increase our revenues; (ii) reduce the time to market; and (iii) contribute to maintain at least one full time job at Open Engineering.

Proposed high-level Work Plan

Start date:		Q2-2020		
Task	**Title**	**Description**	**SME effort (PM)**	**PRACE effort (PM)**
1	Initial implementation	This task consists in two parts. The first part consists in the selection of a parallelisation strategy. It will be selected in close cooperation with the PRACE partner. The second part consists in the development, by the Open Engineering partner, of a first implementation of a multi-GPUs version of its RF solver.	2.0	0.5
2	Initial Profiling	The software, developed in the previous task, will be handed over to the PRACE partner and installed on the target system, along with example test cases and benchmarks. These will be verified and profiled to identify bottlenecks.	0.5	1.0
3	Optimisation strategy	Based on profiling data from previous task, propose strategies for code optimisation.	0.25	1.0
4	Second implementation	Based on the optimisation strategy proposed during the previous task, development by Open Engineering of a second version of the multi-GPUs version of the RF solver.	1.0	0.5
5	Verification	Verification of the positive impact of the proposed optimisation strategy on the code.	0.25	1.0
6	MPI investigation	Preliminary activities in view of the development of a MPI version of the parallel RF solver.	0.5	0.5
7	Final report	Produce report on the outcomes of the work.	0.5	0.5
		Total	5.0	5.0

Technical and business requirements

Compute Resource

Existing architecture	The current implementation of the code works in parallel. Its OpenMP implementation has been tested on a single processor (Intel core i7) workstation and a dual processor (Intel Xeon) server. Its CUDA implementation has been tested with a single NVIDIA GPGPU (GeForce GTX 1080 A8G GAMING) and a single Tesla card (K40c & V100).
Preferred architecture	Our first goal with this project is to develop a multi-GPU implementation of our CUDA code. We assume that GPUs are all attached to the same host. We would like to get access to a server equipped with several TESLA GPUs from NVIDIA. After this project, we would like to add MPI support to the code to make a hybrid version of it that runs with several nodes each one equipped with several GPUs and/or multiple CPUs. Therefore, by the end of the project, we would like to get access to several nodes (with several GPUs attached to each one) for preliminary investigations.
Parallelisation strategy	In this project, we would like to focus on a multi-GPU version of our CUDA code with discrete GPUs attached to the same host. Only a small time, at the end of the project, will be dedicated to preliminary investigations on an MPI strategy in view of working with several nodes.
Storage (Gbyte)	GPU storage requirements: 32GB (minimum) HDD storage requirements: 1TB
Third party software	In terms of software requirements, our code is written in C++ and in CUDA. Its works under Windows and Linux. Under Linux, our developments are done with Centos 6, we use GCC version 7.3.1 with glibc 2.12 and CUDA 10.1. We use cmake 3 for compilation and Paraview for the visualization.
Typical run	1 hour on a GPU (for instance, in a Tesla V100 with 32GB)
Core hours	For instance, for one node with 4 GPUs it will be required 500hs. Although the total number of hours scales linearly with the number of nodes given.
Memory	Between 128-256GB of main memory per node (assuming 4 GPUs in each node)

In terms of HPC architecture, our goal is to have access to at least one computer host to which several discrete GPUs from NVIDIA are attached.

After this project, our ambition is to continue the effort with the development of a parallel version of the code with MPI to split the computation workload between several hosts each one equipped with multiple GPUs. This is the reason why we have included task 6 (preliminary

MPI investigation) in our work plan and would like to get access to more than one host by the end of the project.

As for now, we have no PRACE centre to mention specifically. This centre shall have access to the desired hardware and, at least, have knowledge in multi GPUs programming with CUDA. Expertise with parallel FDTD is a plus.

Non-technical resource

Confidentiality	Yes

This activity concerns our proprietary code, so an NDA will be needed.