
E-Infrastructures
H2020-INFRAEDI-2018-1

INFRAEDI-02-2018: Centres of Excellence on HPC

# EoCoE-II

Energy Oriented Center of Excellence:
toward exascale for energy

Grant Agreement Number: 824158

## D2.1
## Application starter outcome: preliminary performance evaluation results and first optimization roadmap

## Project and Deliverable Information Sheet

| | | |
|---|---|---|
| EoCoE-II | Project Ref: | EINFRA-824158 |
| | Project Title: | Energy Oriented Center of Excellence: toward exascale for energy |
| | Project Website: | http://www.eocoe2.eu |
| | Deliverable ID: | D2.1 |
| | Deliverable Nature | Report |
| | Dissemination Level*: | PU |
| | Contractual Date of delivery: | M6 |
| | Actual Date of delivery | M6 |
| | EC Project Officer | Andrea Feltrin |
| | | |

* - The dissemination levels are indicated as follows: PU – Public, CO – Confidential, only for members of the consortium (including the Commission Services) CL – Classified, as referred to in Commission Decision 2991/844/EC.

## Document Control Sheet

| | | |
|---|---|---|
| Document | Title: | Application starter outcome: preliminary performance evaluation results and first optimization roadmap |
| | ID: | D2.1 |
| | Available at: | http://www.eocoe2.eu |
| | Software Tool: | Microsoft Word |
| Authorship | Written by: | Mathieu Lobet, PhD (CEA) |
| | Contributors: | Bibi Naz, PhD (FZJ)<br>Edoardo Di Napoli, PhD (FZJ)<br>Georg Hager, PhD (FAU)<br>Gerhard Wellein, PhD (FAU)<br>Hendrik Elbern, PhD (RWTH)<br>Herbert Owen, PhD (BSC)<br>Jan Eitzinger, PhD (FAU)<br>Johanna Bruckmann, PhD (RWTH)<br>Jose Fonseca, PhD (CEA)<br>Julien Thélot (CEA)<br>Philipp Franke, PhD (FZJ)<br>Ulrich Ruede, PhD (FAU)<br>Stefan Kollet, PhD (FZJ) |

| | | |
|---|---|---|
| | | Yanick Sarazin, PhD (CEA) |
| | Reviewed by: | PEC, PSB |

Document Keywords: programming model, optimization, Exascale, CPU, GPU, performance Web platform; Internet web site

## Executive Summary

## Index

# Acronyms and abbreviations

**AMR**: Adaptive Mesh Refinement
**BSC**: Barcelona Supercomputing Center
**CEA**: Commissariat à l'énergie atomique et aux énergies alternatives
**CoE**: Center of Excellence
**CPU**: Central Processing Units
**FAU**: Friedrich-Alexander University of Erlangen-Nuremberg
**FZJ**: Forschungszentrum Jülich GMBH
**GPU**: Graphical Processing Unit
**HPC**: High Performance Computing
**IFPEN**: IFP Énergies nouvelles
**LLNL**: Lawrence Livermore National Laboratory
**MDLS**: Maison de la Simulation
**NREL**: National Renewable Energy Laboratory
**PDAF**: Parallel Data Assimilation Framework
**PDI**: Parallel Data Interface
**POP**: Performance Optimization and Productivity CoE
**R-CCS**: RIKEN Center for Computational Science
**RWTH**: Rheinisch-Westfälische Technische Hochschule Aachen, Aachen University
**WP**: Work Package

# Introduction

The purpose of this document is to summarize the start of the project in month six and to present an updated work plan for the rest of the project regarding activities within WP2. The general objective of WP2 on programming models is to address HPC performance, parallelism and code scalability at different levels, code architecture and technology limitations. The goal consists on optimizing the selected applications to be able to run challenging scientific scenarios. For some of them, we want to make them ready for pre-exascale or Exascale ecosystem. As defined in the proposal, we have divided the WP6 objectives into six points:

- Performance evaluation
- Application performance efficiency
- Application platform flexibility
- Application code readability
- HPC tools and libraries knowledge
- HPC tools and libraries improvement

The following part of this document is divided into 6 parts that correspond to the tasks in WP2. The first one is the transversal activities and the following ones correspond to the different SC.

# Performance evaluation and modeling (Task 2.1)

## 1. Performance evaluation and optimization workshops

Workshops will play a major role in achieving the goal of WP2 during the project. They will enable to:

- Evaluate application performance issues
- Teach application teams to use performance evaluation tools and recognize performance bottlenecks
- Identify optimization opportunities and guide optimization strategies
- Initiate collaborations with the performance engineering experts
- Follow performance and optimization progress

Workshops will be organized with the High-Performance Computing division of the Friedrich-Alexander University of Erlangen-Nuremberg (FAU). This partner constitutes the main performance and optimization expertise within WP2. Workshops will also involve the Performance Optimization and Productivity (POP) Center of Excellence (lien vers le site POP). Both entities are strongly involved in the workshop organization.

Workshops are divided into 2 categories:

- Performance evaluation workshop
- Hackathon workshop

We will first organize a Performance Evaluation Workshop. This first session of 3-day will help prepare the hackathons. During this first session, application teams will learn the basics of performance engineering (mainly at node level). They will understand how to use performance evaluation tools from FAU and PoP and how to investigate performance bottlenecks. Application developers will test the performance evaluation tools with their codes. This first session will ensure that all code developers, and particularly developers involved in code refactoring and optimization, are on the same level of knowledge. After attending the workshop, application developers should have the means to start initial performance analysis on their own. One stated goal of the workshop is also to connect HPC experts with application developers within the project, which will help tremendously to develop a common vocabulary and understand each other's workflow. It is supposed to initiate point-to-point collaboration between experts and scientific teams if not already started before the workshop.

The next sessions will have a different format commonly called "hackathon." Thanks to the performance evaluation results, application developers will have guidelines to start optimizing their codes on their own. Hackathons will gather HPC experts and application developers to work on specific optimization issues during approximately 3-day several times during the project. They will enable to overcome strong performance bottlenecks or complex optimization challenges for application developers. Hackathons will also help to track the optimization progress and update performance-aware code development strategies.

**Workshop agenda:**

Meetings between HPC experts and the WP2 leader concerning workshop organization are organized every month or bimonthly depending on the needs. The first workshop was supposed to be held before M6 (from the 3rd to the 6th of June 2019) but was cancelled, mainly due to lack of participants. This did not come as a

surprise because the workshop was organized during a busy season, and many scientific teams had not yet hired developers to work on optimization aspects.

Table 1 shows the tentative WP2 workshop agenda. We plan to organize the Performance Evaluation Workshop in October 2019 (M9-10). Then, we plan to have the first hackathon 6 months later in February-March 2020 (M14-15). Another hackathon could be organized around September 2020.

Besides workshop preparations, preliminary benchmarking activities have been started at FAU in order to fathom the suitability of the ARM-based Cavium/Marvell ThunderX2 processor for the workloads under investigation in EoCoE-II. A comparative analysis paper is currently under review. ARM systems are in the focus of interest in the community because the European Processor Initiative (EPI) has chosen ARM as one of its underlying architectures.

| | Project Month | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Workshop name** | **6** | **8** | **10** | **12** | **14** | **16** | **18** | **20** | **22** | **24** | **26** | **28** | **30** | **32** | **34** | **36** |
| Performance evaluation workshop | | | | | | | | | | | | | | | | |
| First hackathons | | | | | | | | | | | | | | | | |
| Second hackathons | | | | | | | | | | | | | | | | |

*Table 1 – Tentative workshop agenda*

## 2. Collaboration between experts and application developers

The following Table 2 presents HPC experts and their roles within the project.

| People | Position | Role | Period |
|---|---|---|---|
| Georg Hager | FAU | Node-level optimization | M1-M36 |
| Gerhard Wellein | FAU | Coordinator | M1-M36 |
| Jan Eitzinger | FAU | Node-level optimization | M1-M36 |
| Thierry Gautier | CR INRIA Equipe Projet INRIA AVALON - ENS Lyon | Expert in task-based programming model. Originally integrated in the project to support the refactoring of Gysela, his general expertise in parallel programming can be useful for all members. | M1-M36 |
| Judit Gimenez | Member of the POP COE | HPC Experts and lecturer for the POP trainings | Workshops |
| Brian Wylie | Member of the POP COE | HPC experts and coordinator for POP | Workshops |

*Table 2 - HPC experts in WP2*

We will develop point-to-point collaboration between HPC experts and application developers. Workshops will help to set these collaborations up. Some of them have already started and are presented in Table 3:

| Code | State |
|---|---|
| Alya (Wind SC) | Has recently contacted FAU to initiate a collaboration. They have also HPC experts locally at BSC. |
| WalBerla (Wind SC) | WalBerla is developed at FAU and will benefit from the local expertise |
| PVnegf (Material SC) | Is already in close contact with FAU for node-level optimization |

*Table 3 – Collaboration with HPC experts*

# Wind code optimization (Task 2.2)

WP2 involves the flagship code Alya and the satellite codes WalBerla and SOWFA. Figure 1 schematically describes the task structure.

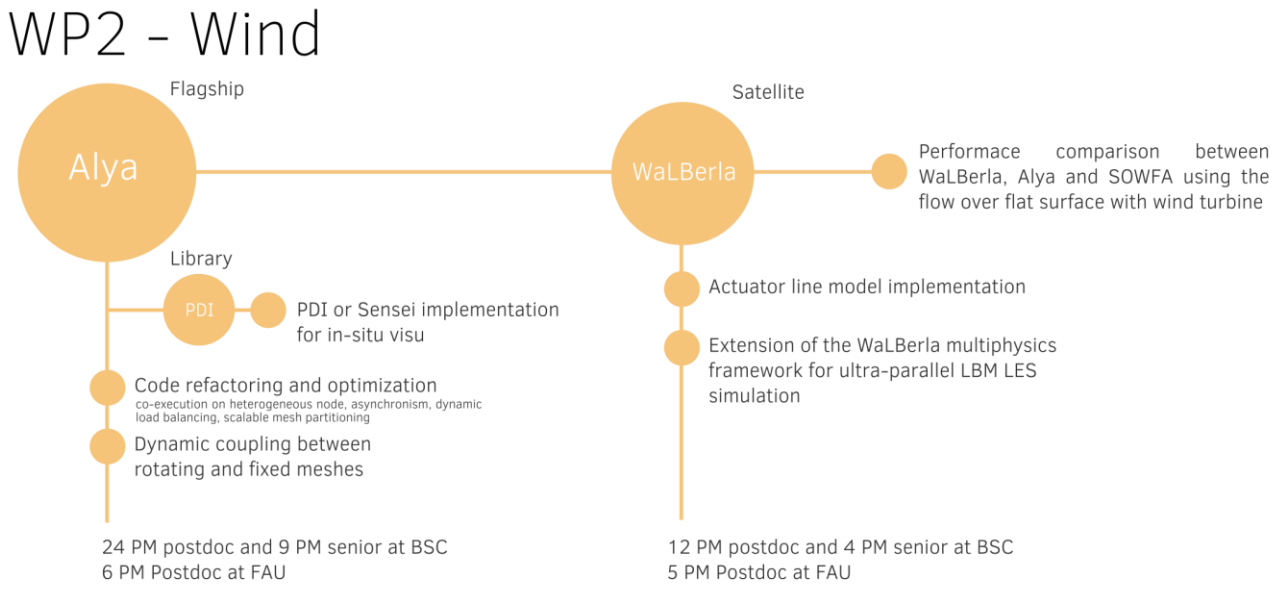


*Figure 1 - Task structure for the Wind activity within Work Package 2*

## 1. Flagship code Alya

Alya is a high-performance computational mechanics code that solves complex coupled multi-physics problems, mostly coming from the engineering realm. BSC develops this code (Alya website).

The main goal is to bring the flagship Alya code to Exascale in order to tackle the simulation of full wind farm over complex terrain with up to 100 wind turbines. Within WP2, Alya's developers with HPC experts will refactor and optimize the code to be able to address heterogeneous computing nodes with maximal efficiency. They will implement a full rotor model where the actual geometry of the wind turbine is modelled.

**Recruitment and people:**

BSC will use the allocated PMs to maintain the current staff on the code. Table 4 presents the main developers of the code.

| People | Position | Role | Period |
|---|---|---|---|
| Herbert Owen, PhD | Senior researcher at BSC | Responsible for the Alya team within EoCoE and developer of the code | M1-M36 |
| Guillaume Houzeaux, PhD | Physical and numerical group manager at BSC | Main Code developer | M1-M36 |

*Table 4 – Members of the Alya team in WP2*

**Co-execution on heterogeneous cluster with scalable geometric mesh partitioning:**

BSC has already made some progress on co-execution on heterogeneous clusters (CPU + Accelerators). The Alya code has been adapted so that it can work, not only on CPUs, but also on GPUs for Computational Fluid Dynamics problems, chiefly Large Eddy Simulation cases. For such problems, a semi-implicit approach is used where the momentum equation is solved explicitly while the continuity equation is solved implicitly. The pressure matrix remains constant during all of the simulation, which involves tens of thousands of time steps. Thus, the computation time for its creation is negligible. Using a fractional step scheme, the two most expensive kernels are, therefore, the calculation of the right-hand side vector for the momentum equation and the solution of a linear system for the pressure at each time step. For the former kernel, OpenACC has been used to adapt the code to GPUs while for the later Alya's own linear solvers have been ported to CUDA. We are currently working in WP3 to use EoCoE-II-provided linear solvers that can run on GPU's.
Taking advantage of the fact that most of Alya can now run either on CPUs or GPUs, we have decided to develop a co-execution approach that makes better use of current pre-exascale supercomputers, which typically blend GPUs and CPUs. In this way, we make full usage not only of the GPU's but also of the CPUs, which are usually underused in such machines. A fast and scalable geometric mesh partitioning based on Space Filling Curve (SFC) has been key to enable the co-execution with a correct load balance between the GPU's and CPU's. At the beginning of the simulation, the SFC partitioning is called iteratively several times until an optimum partitioning of the mesh is obtained.
In the first iteration, each MPI task (be it CPU or GPU) receives a certain portion of the mesh according to some initial weights. With this partition, it calculates a couple of time steps. Based on the computational time taken by each MPI task, it adapts the weights and repartitions again. After a couple of iterations, each processor receives the correct amount of work so that they all take nearly the same time. GPUs obviously receive a bigger chunk of the mesh than CPUs. In this way, the work done by the CPUs is spared in comparison to a pure GPU calculation.
Tests have been performed on the MareNostrum POWER9 supercomputer formed by three racks of last IBM POWER technologies (POWER9 CPUs plus Volta GPUs) with a peak performance of 1.5 petaflops. A complex geometry problem discretized with 176 million finite elements has been used. The use of co-execution has allowed reducing the computational time by approximately 20 %, if compared with a pure GPU calculation.

**Dynamic coupling between rotating and fixed meshes:**

Alya counts with a parallel coupling library that allows to couple Alya to other codes. It also allows to couple two or more instances of Alya. This can be used, for example, to solve Fluid-Structure Interaction problems where one Alya solves for the Fluid part and the other one for Solid part while the coupling allows to interchange forces and displacements between both instances. The coupling library can also be used for problems in which one part of the domain rotates while the other one is fixed using a different instance of Alya on each part. Preliminary testing of the methodology for incompressible flow problems with complex geometry (a rotating car wheel) has provided positive results. Some robustness issues have been identified, and we are currently working on them. As soon as the robustness issues are solved, testing of a rotating NREL Phase VI wind turbine will start.

**In-situ visualization tool integration**

The integration of PDI or Sensei is not yet started. The choice of one of these tools is not yet done.

**Comparison with SOWFA:**

BSC has shared some benchmarks with IFPEN that they will start running with SOWFA.

**Alya updated project schedule:**

Alya's developers are currently working on three subtasks at the same time: co-execution, mesh partitioning and dynamic coupling. They give the priority to these tasks before starting the dynamic load balancing and asynchronism (communication and computation overlapping). They will address general code optimization that includes code cleaning, node-level optimization and vectorization all along the project. Table 5 illustrates graphically the work plan.

| | Project Month | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Alya sub-tasks** | **6** | **8** | **10** | **12** | **14** | **16** | **18** | **20** | **22** | **24** | **26** | **28** | **30** | **32** | **34** | **36** |
| Alya general code optimization | | | | | | | | | | | | | | | | |
| Dynamic load balancing | | | | | | | | | | | | | | | | |
| MPI overlapping between communication and computation | | | | | | | | | | | | | | | | |
| Co-execution on heterogeneous cluster | | | | | | | | | | | | | | | | |
| Fast and scalable geometric mesh partitioning | | | | | | | | | | | | | | | | |
| Dynamic coupling between rotating and fixed meshes | | | | | | | | | | | | | | | | |
| Scaling on exascale or pre-exascale machines | | | | | | | | | | | | | | | | |
| Performance comparison with WalBerla and SOWFA | | | | | | | | | | | | | | | | |

*Table 5 - Alya updated work plan*

## 2. Satellite code WalBerla

WalBerla is a fluid simulation code that uses the lattice Boltzmann method (WalBerla website). WalBerla is developed at the Friedrich-Alexander University of Erlangen-Nuremberg (FAU). In WP2, WalBerla developers will implement an actuator line model. The final goal is to be able to simulate wind turbine with the lattice Boltzmann method and to compare the results with the flagship code Alya and the code SOWFA.

**Recruitment and people:**

FAU is coordinating with IFPEN to hire a candidate in September 2019. The following table, Table 6, summarizes People involved in the project for WalBerla:

| People | Position | Role | Period |
|---|---|---|---|
| Ulrich Ruede, PhD | FAU | Responsible for the WalBerla code | M1-M36 |
| Under recruitment | Postdoc at FAU with IFPEN | Code optimization and development | M9-M33 |

*Table 6 – Members of the WalBerla team within WP2.*

**Work progress:**

The implementation has not yet started. First meetings will be held in July 2019 and the real work will start with the recruitment.

**WalBerla updated project plan:**

WalBerla team estimates that they need approximately 3 months to prepare the code to the simulation of wind turbine, then 6 months to integrate the actuator line model to get the first performance and simulation results with a single turbine the following 3 months. In term of development, the remaining part of the project could be spent on extending the WalBerla models to simulate wind farms. This requires the implementation of a load balancing and gird refinement capability that is possible only with a close collaboration with IFPEN. At the same time, the remaining part of the project will be spent on the comparison with Alya. Table 7 illustrates the updated the work plan.

| | Project Month | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WalBerla sub-tasks | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 |
| WalBerla code preparation for wind turbine | | | | | | | | | | | | | | | | |
| Integration of the actuator line model | | | | | | | | | | | | | | | | |
| First performance results on a single wind turbine | | | | | | | | | | | | | | | | |
| Comparison with Alya | | | | | | | | | | | | | | | | |

| | |
|---|---|
| Extending the waLBerla models from a single wind turbine to wind farms would be desirable | |

*Table 7 – WalBerla updated work plan*

# Meteorology code optimization (Task 2.3)

EURAD-IM is the only flagship code concerned by the WP2. Figure 2 schematically shows the sub-tasks structure.

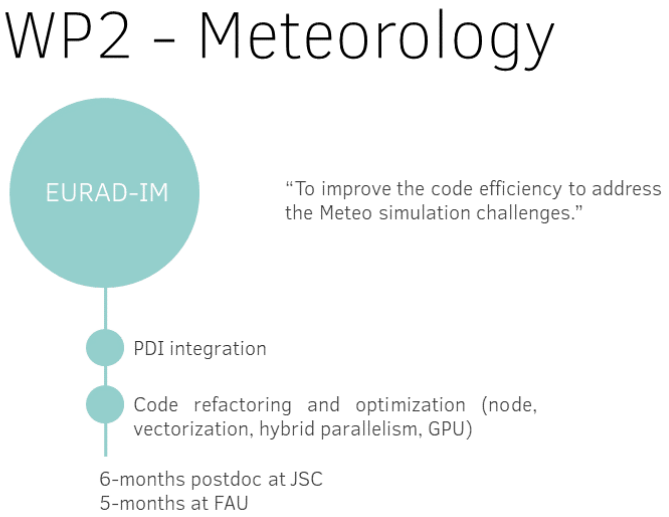


*Figure 2 - Task structure for the Meteorology activity within Work Package 2*

## EURAD-IM

EURAD-IM simulates chemistry particle transportation in local atmospheres coupled with a weather forecast application WRF. An advection-diffusion-reaction equation, with multiple solvers for chemistry, is used. In EURAD-IM, the stiff gas phase solver is one of the main performance bottlenecks and most time consuming part. The objective of WP2 is to improve the code efficiency to address the following Meteo simulation challenges with main items:

- PDI integration (with CEA PDI expert) for IO optimization in WP4 and ensemble runs in WP5,
- Code refactoring (with FAU) including change of data structure for vectorization and memory management,
- Node level optimization (with FAU) and vectorization of the stiff gas phase ODE solver, AND
- Hybrid parallelization MPI + OpenMP/OpenACC to improve the parallelization on large-scale CPU machine first and leverage the possibility of GPU usage.

**Recruitment and people:**

The recruiting process is ongoing. Applicants have been interviewed and finally the position was offered to a candidate, with acceptance pending. The candidate will work in WP2 and WP4 on PDI.

| People | Position | Role | Period |
|---|---|---|---|
| Hendrik Elbern, PhD | Senior scientist at RWTH | Scientific coordinator | M1-M36 |
| Philipp Franck, PhD | Postdoc at FZJ | EURAD-IM code expert | M1-M36 |

*Table 8 – Members of the EURAD-IM team within WP2*

**Work progress**

On a first evaluation, the ODE solver for chemical processing was identified as the main computing time-consuming part of EURAD-IM. Together with experts from FAU the specific routines were extracted and investigated for possible performance bottlenecks. Thus, index allocation of arrays were identified, which limit the node-level optimization. Further investigations of the specific codes are required and planned for possible solutions to this bottleneck.

**EURAD-IM updated work plan:**

The work plan for EURAD-IM is presented in Table 9.

| | Project Month | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **EURAD-IM sub-tasks** | **6** | **8** | **10** | **12** | **14** | **16** | **18** | **20** | **22** | **24** | **26** | **28** | **30** | **32** | **34** | **36** |
| PDI | | | | | | | | | | | | | | | | |
| Code refactoring (node) | | | | | | | | | | | | | | | | |
| Code refactoring (vectorization) | | | | | | | | | | | | | | | | |
| Code refactoring (hybrid parallelism) | | | | | | | | | | | | | | | | |
| Code refactoring (GPU) | | | | | | | | | | | | | | | | |

*Table 9 – EURAD-IM updated work plan*

# Materials code optimization (Task 2.4)

PVnegf is the only flagship code concerned by the WP2. Figure 3 schematically shows the sub-tasks structure.

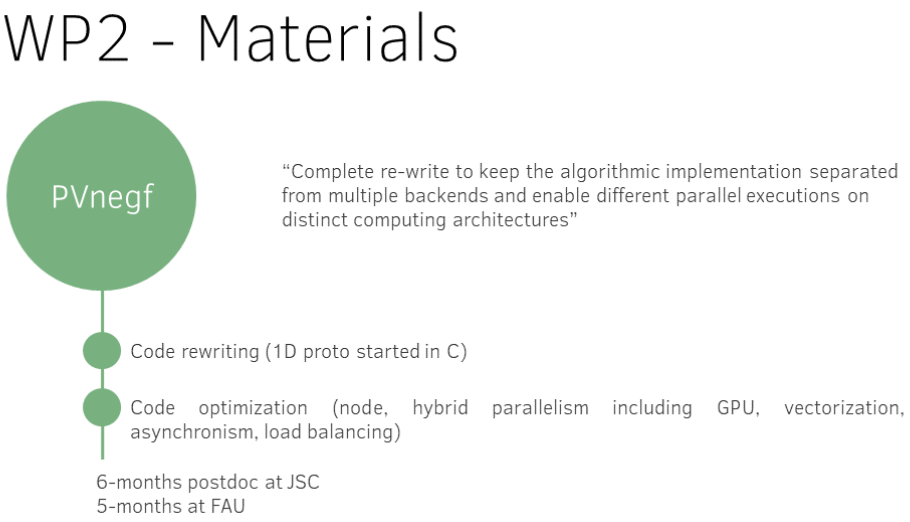


*Figure 3 - Task structure for the Materials activity within WP2*

## Flagship code PVnegf

PVnegf provides photo carrier dynamics (generation, transport and recombination) of nanostructured regions and at complex interfaces in advanced high-efficiency solar cell devices. It solves the steady-state non-equilibrium Green's function - Poisson equations for charge carriers coupled to photons and phonons. Both interactions are treated on the level of self-consistent Born self-energies. The code can be coupled to a wide range of codes for electronic and vibrational structures, from simple effective mass or continuum elasticity pictures to empirical and ab initio tight-binding and force-field approaches.

**Recruitment and people:**

Table 10 shows people involved in the WP2.

| People | Position | Role | Period |
|---|---|---|---|
| Edoardo di Napoli, PhD | Senior scientist at the Jülich Research Center (Forschungszentrum Jülich – FZJ), | Supervises and coordinates the PVnegf activity | M1 – M36 |
| Paul Baumeister, PhD | Senior scientist at FZJ | Participates to the coordination of the work and is involved in the code refactoring | M1 – M36 |
| Sebastian Achilles | PhD student at FZJ | In charge of the refactoring and the parallelization | M1 - M21 |
| Georg Hager | Senior scientist at FAU | Expertise and advisor in code optimization and HPC | M9-M15 |

*Table 10 – Members of the PVnegf team within WP2*

**Code refactoring**

The refactored code will assume the new name neXGf. Besides the obvious objective of having a high-parallel and scalable code, the neXGf project aims at:

- Clarity: a small code that is readable and extensible;
- Ease-of-use: error detection and communication, high-quality tutorials and Continuous Integration.

The PVnegf refactoring will be carried out in C++, starting from the current Fortran 90 code. The final code will have two levels of parallelization: MPI + OpenMP with vectorization on CPU and MPI + OpenACC on GPUs (which can eventually be replaced by OpenMP for GPU when mature). Kokkos is a possible way to handle the cross-platform implementation.

The re-written version will use the experience acquired with 1Dnegf. 1Dnegf is an application developed in C by Sebastian Achilles that uses the basic functionalities of PVnegf. It is a fully paralyzed code based on MPI + OpenMP with a good scalability (78%) up to 1.8 million cores.

**PDI integration**

The I/O operation within PVnegf are quite limited and we do not foresee the use of PDI in this context. Nonetheless, once the new code neXGf will assume its final fully parallelized form, PDI integration would be implemented with the intention of writing and reading intermediate (and possibly large) amount of data for checkpointing in crucial steps of the two iterative nested loops at the base of PVnegf algorithm.

**Project management:**

A simplified Scrum workflow will be used to plan and monitor progress of the project especially in its advanced phase (Milestones 4 and 5) where the entire team including members from WP1, 2, 3 and 4 will participate in the implementation, testing and validation.

**PVnegf/neXGf updated project schedule:**

In the first part of the project, the refactoring of PVnegf will focus on rewriting the current validated functionalities of PVnegf with a clean and modern approach. The rewriting will then proceed in an orthogonal direction, focusing on the parallelization and optimization of the re-written code. Figure 4 shows in a schematic manner the nexGf roadmap.

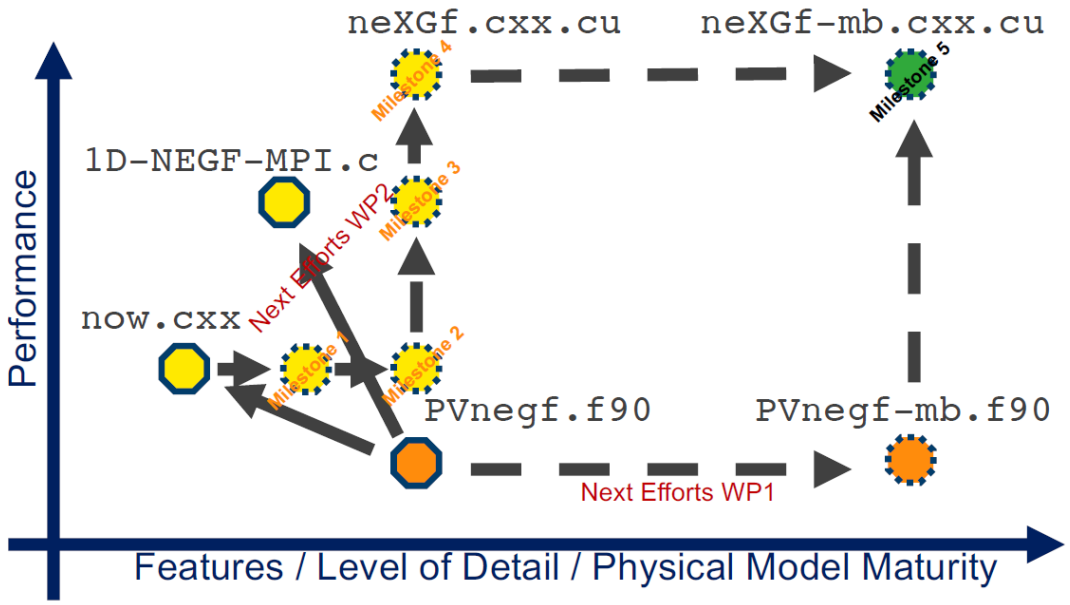


*Figure 4 – description of the neXGf development roadmap.*

The WP2 work can be divided into the following subtasks:

- Milestone 1: Rewriting of the non-ballistic code including electrons and holes interaction with phonons. The code will be only node-level parallel using OpenMP directives. Testing units will be included together with a first basic version of Continuous Integration (CI) within a private Gitlab repository. First optimization at the node-level will be carried out at this stage in collaboration with FAU.
- Milestone 2: All the major functionality of PVnegf at the current stage (without multi-bands) will be included in the new re-written code, including photons interaction but excluding irregular grids for the energy grid points and minor functionalities such as executing simulation with only one type of charge transport. At this stage, the new code will be verified against PVnegf using a number of validated input files.
- Milestone 3: Full parallel code with two level of MPI together with the node-level parallelization through OpenMP. The inclusion of an interface, which could make use of the Kokkos library, is an optional task. PDI for the checkpointing will be integrated at this stage. While fully parallel, the refactored code will not be yet fully tuned and optimized. Specialized linear algebra kernel will be integrated and further tested at this stage in collaboration with Inria.
- Milestone 4: At this stage the refactored code, neXGf, will be fully optimized with respect to load balancing, overlapping between communication and computation, vectorization over CPU cores and acceleration over multiple GPUs per node. Irregular grids for the energy and simulation corner cases will also be part of this milestone. neXGf code will have a full-fledged CI mechanism, unit testing and will be tested and verified on the largest pre-exascale platforms available.
- Milestone 5: While the PVnegf code is refactored into neXGf, PVnegf will also undergo an evolution, bringing it from a single band quasi 1D code to a multi band quasi 1D code. All the changes entailed by this evolution will drastically change the data structure and will require full validation. From M21 to M30 we plan to integrate such scientific evolution in the fully parallelized and optimized neXGf. Since this step will require a complete change in the data structure, its delivery may encounter unexpected difficulties and delays. For this reason, this milestone comes quite late in the project

while maintaining a 6-month window to deal with unforeseen issues. The scientific payload that is part of WP1 will also be delivered within this stage in collaboration with partners from ENEA.

The corresponding updated work plan is presented in Table 11.

| | Project Month | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **PVnegf sub-tasks** | **6** | **8** | **10** | **12** | **14** | **16** | **18** | **20** | **22** | **24** | **26** | **28** | **30** | **32** | **34** | **36** |
| First rewriting of the code | | | | | | | | | | | | | | | | |
| Migration of the major functionality of Pvnegf in the re-written version of the code neXGf. | | | | | | | | | | | | | | | | |
| Fully parallelized version of the code | | | | | | | | | | | | | | | | |
| Fully optimized version of the code | | | | | | | | | | | | | | | | |
| Multi-band quasi-1d code | | | | | | | | | | | | | | | | |

*Table 11 - Pvnegf/neXGf updated work plan*

## Hydrology code optimization (Task 2.5)

The flagship codes ParFlow and SHEMAT-Suite are both involved in WP2. Figure 5 schematically shows the sub-tasks structure.

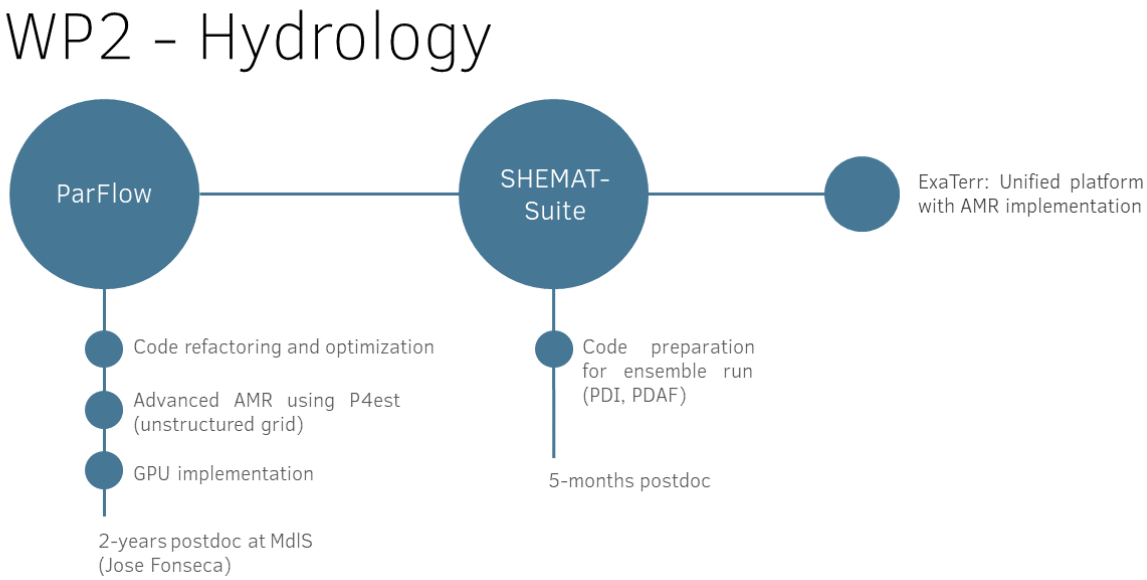


*Figure 5 - Task structure for the Materials activity within WP2*

## 1. Flagship code ParFlow

ParFlow is a parallel, integrated hydrologic model, which simulates surface and subsurface flow. It is based on the shallow water equations coupled with the three dimensional Richard's equation. The code provides a solver for the latter based on a cell-centered finite difference scheme on regular Cartesian meshes. Time integration is performed with an implicit Euler method. The resulting system of nonlinear algebraic equations is solved by a multgrid-preconditioned Newton-Krylov method.

Within WP2, we can divide the work in ParFlow into 3 points:

- Modernization of the code
- Activation of the Adaptive Mesh Refinement (AMR) of the computing grids with the ad hoc numerical scheme and corresponding solver using the library p4est
- GPU implementation

**Recruitment and people:**

Maison de la Simulation, CEA has hired a HPC specialist to work on ParFlow, Jose Fonseca, in May 2019 (M5). He was already working in ParFlow in EoCoE-I, on code modernization, general optimization and AMR aspects. FZJ will recruit a computer scientist in the IBG division to work on optimization aspects as well. Table 12 summarizes the people involved in WP2.

| People | Position | Role | Period |
|---|---|---|---|
| Stefan Kollet, PhD | FZJ | Scientific coordinator | M1-M36 |
| Bibi Naz, PhD | FZJ | PDI aspect | M1-M36 |
| Mathieu Lobet, PhD | Research-engineer at MdlS, CEA | HPC coordinator at CEA, MdlS | M1-M36 |
| Jose Fonseca, PhD | Postdoc at MdlS, CEA | HPC expert that will develop AMR aspects of ParFlow and GPU aspects | M5 – M29 |
| Under recruitment | FZJ | Computer Scientist that will focus on code optimization and GPU aspects | M9 – M33 |

*Table 12 – Members of the ParFlow team in WP2*

**Performance results:**

FZJ has performed strong scaling tests at the beginning of the project on the Juwels supercomputer located at Jülich. They have used different compilers to confirm the results. They have also repeated the scaling study with different muligrid-based preconditioners:

- PFMG provided by the external dependency HYPRE
- PFMG Octree, which is essentially a memory-optimized version of PFMG and builds the preconditioner matrix using internal ParFlow's octree structures.

- MGSemi, which is a preconditioner coded inside ParFlow

In Figure 6, we show the scaling results for the preconditioners themselves. It reveals that HYPRE preconditioners provide bad scaling numbers in comparison with MGSemi. The cause of this issue is still under exploration. Our current hypothesis is that it may be due to the version of HYPRE.

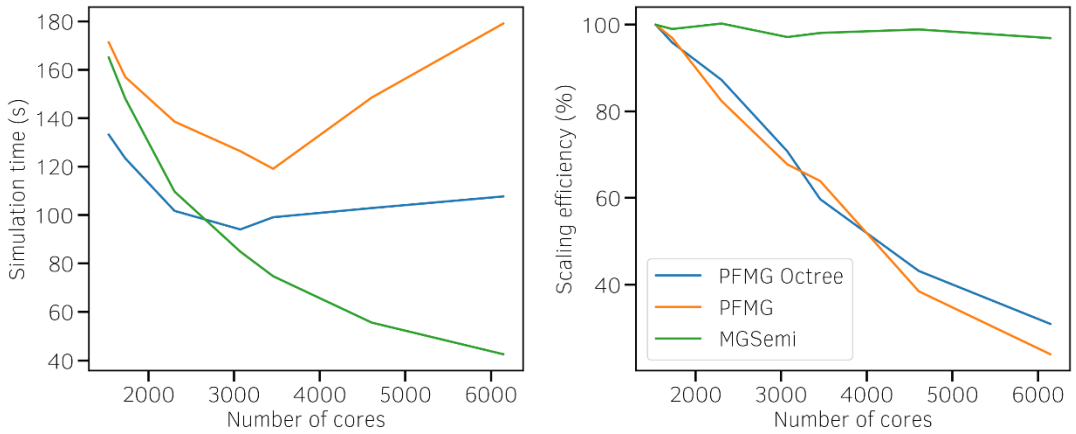


*Figure 6 - Results of a strong scaling study on the Juwels machine at Jülich. On the left, the computational time for the the PFMG (blue and orange) and the MGSemi (green) preconditioners. On the right, the scaling efficiency in percentage for the same components.*

Some optimization efforts are also ongoing in USA laboratories at the same time. Preliminary performance analysis demonstrates that the octree structure may be a performance bottleneck (LLNL developers). They have started to test the GPU implementation of the HYPRE library as well with reported speedups of 7x using the CUDA PFMG solver.

**Adaptive Mesh refinement (AMR) implementation:**

A project leaded by Carsten Burstedde at the University of Bonn and Stefan Kollet at FZJ, funded by the German Research Fundation (DFG) has led to the integration of ParFlow and p4est. Currently, it is possible to use the latter as the mesh manager of the former in the context of uniform meshes, and hence, without explicitly exploiting the AMR capabilities provided by the p4est library.

The objective is to extend the existing ParFlow's integration with p4est by activating the refinement and allowing the use of locally refined meshes. This task requires several changes in the ParFlow core code that we have divided into the following subtasks:

- Correctly propagate in ParFlow the neighboring information provided by p4est for the case of statically refined meshes.
- Reinterpretation of ParFlow's native finite difference scheme as a mixed finite
- Element method (MFE) with Raviart-Thomas spaces of low order.
- Investigate how the fluxes should be interpolated at the interface of two different size mesh elements.
- Implement a preconditioned Krylov saddle point solver for MFE in ParFlow.
- Study a posteriori error indicators for the Richards' equation.

The expected time spam to work on these sub-tasks is presented in the corresponding updated work plan displayed in Table 13.

**GPU implementation:**

The GPU adaptation of the code is part of the code modernization. As a first step, ParFlow team will adapt to GPU the preconditioners and the solvers. They are currently under discussion with the developers of the scalable linear algebra library HYPRE developed at the LLNL in the United States. HYPRE has now a GPU implementation with different backend (Cuda, RAJA, kokkos). ParFlow uses HYPRE and some parts of the code are directly adaptable from HYPRE (similar implementations). The GPU porting may be performed at the beginning of the project via this collaboration between the USA (Laura Condon, University of Arizona and her PhD student) and the newly hired computer scientist at Jülich.

The second step is the full conversion of the code including the AMR aspects. ExaTerr sub-tasks may cover this aspect.

**ParFlow updated work plan:**

Table 13 summarizes the updated work plan for ParFlow.

| | Project Month | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ParFlow sub-tasks | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 |
| Correctly propagate in ParFlow the neighboring information provided by p4est for the case of statically refined meshes | | | | | | | | | | | | | | | | |
| Reinterpretation of ParFlow's native finite difference scheme as a mixed finite element method with Raviart-Thomas spaces of low order | | | | | | | | | | | | | | | | |
| Investigate how the fluxes should be interpolated at the interface of two different size subgrids | | | | | | | | | | | | | | | | |
| Port the saddle point multigrid preconditioner into ParFlow | | | | | | | | | | | | | | | | |
| Study a posteriori error indicators for the Richards' equation | | | | | | | | | | | | | | | | |
| GPU implementation of the solvers | | | | | | | | | | | | | | | | |
| Refactoring of the ParFlow plateform for modularity | | | | | | | | | | | | | | | | |

*Table 13 - ParFlow updated work plan*

## 2. Flagship code SHEMTA-Suite

SHEMAT-Suite is a code for simulating single- or multi-phase heat and mass transport in porous media. It solves coupled problems including heat transfer, fluid flow, and species transport. SHEMAT-Suite can be applied to a range of hydrothermal or hydrogeological problems, be it forward or inverse problems.

**Recruitment and people:**

RWTH Aachen University (GGE) will hire one postdoc researcher for 13 person months to work on various tasks within WP2, WP3, and WP4. The recruiting process has been successful and the researcher will start October 1st, 2019. Five person months are dedicated to WP2. Table 14 presents the members of the SHEMAT-Suite team within EoCoE.

| People | Position | Role | Period |
|---|---|---|---|
| Johanna Bruckmann | RWTH Aachen University | SHEMAT-Suite coordinator; scientist for WP1 | M1-M36 |
| Postdoc under recruitment | RWTH Aachen University | The postdoc will work on WP2, WP3 and WP4 tasks. | M10 – M23 |

*Table 14 – SHEMAT-Suite members within EoCoE-II*

**SHEMAT-Suite updated work plan:**

Work on PDI and PDAF integration to SHEMAT-Suite will start when the respective researcher is hired. We plan to complete the work within 13 person months. A detailed roadmap and task update will only be available when the hired researcher will have started working on the tasks.
The PDI and PDAF activities are shared with WP4. The goal is to optimize the performance of ensemble runs (WP5) for stochastic parameter estimation and uncertainty quantification within geothermal reservoirs.

| | Project Month | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SHEMAT-Suite sub-tasks | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 |
| PDI integration | | | | | | | | | | | | | | | | |
| PDAF | | | | | | | | | | | | | | | | |

*Table 15 – SHEMAT-suite general roadmap, to be updated when the postdoc is hired.*

## 3. ExaTerr platform

ExaTerr is a from-scratch development that aims at building a common software platform for both SHEMAT-Suite and ParFlow. This platform will be based on Kokkos, a software technology strongly pushed by the US DoE, which holds at its core performance portability. The idea pursued here is to have a single code base that could be executed on both CPU and GPU. This platform should handle the AMR aspect under development in ParFlow.

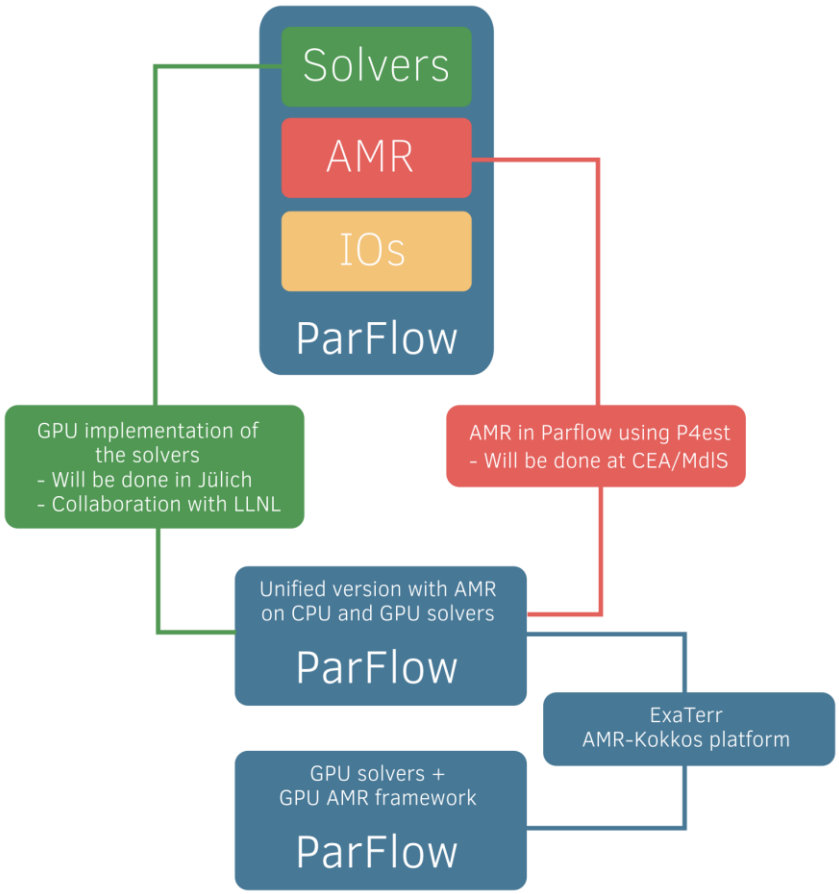


*Figure 7 – Description of the ParFlow and ExaTerr development plan.*

Figure 7 shows the work structure for ParFlow and the ExaTerr platform. At first, in separate branches, ParFlow should have solvers working on GPUs and full AMR capabilities using P4est on CPUs. These two branches should then be merged to test the performance of the following strategy: solvers on GPU with AMR on CPU. This combination may prove inefficient due to the memory transfers from the GPUs and the CPUs. The ExaTerr platform could be then a solution to this issue by deporting some AMR functions directly on the GPU to minimize the memory transfers.

Taking into account the difficulty to reach a working AMR version of ParFlow using P4est with GPU solvers, the development of an ExaTerr porotype will start during the last year of the project.

## Fusion code optimization (Task 2.6)

The flagship code GYSELA-X is the only one represented in the WP2. The original task structure for GYSELA-X from the project proposal is given in Figure 8.

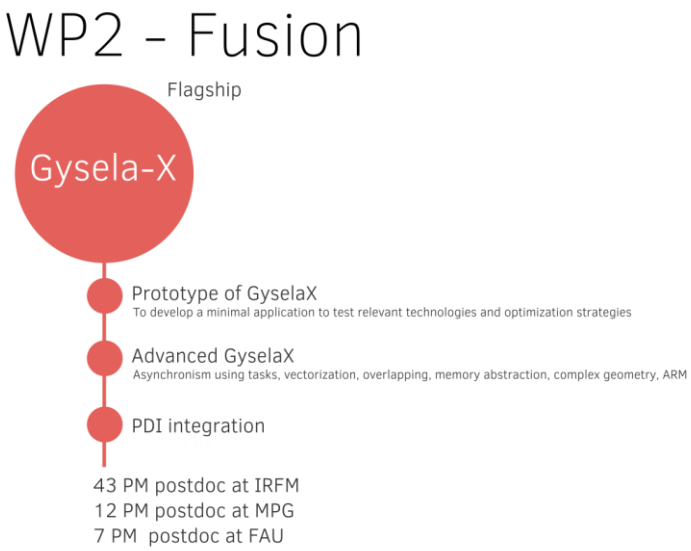


*Figure 8 – Task structure of the Fusion scientific challenge.*

## Flagship code GYSELA-X

GYSELA is a 5D full-f (regarding Vlasov eq.) and flux-driven gyrokinetic Fortran parallel code that solves Vlasov (ions and electrons) and Poisson (electric potential) equations to simulate electrostatic plasma turbulence and transport in the core of Tokamak devices. During EoCoE-II, it will progressively evolve towards the upgraded version GYSELA-X, targeting exascale supercomputers and solving electromagnetic turbulence from the core to the far edge region in ITER-relevant magnetic geometry.

**Recruitment and people:**

The postdocs under recruitment will work on some of the subtasks of Task T2.6.2. Their work is detailed hereafter.

Concerning the third postdoc dedicated to the code optimization in ARM-based clusters, there is a undergoing discussion on the possibility to collaborate with the R-CCS. A joint recruitment could be done with resources coming from both structures for a period of 2 years. This collaboration would take place in the framework of a partnership between CEA and R-CCS under deployment. This would enable to benefit from the Japanese expertise in ARM technologies for HPC and make Gysela-X run on the forthcoming exascale version of the K-computer.

| People | Position | Role | Period |
|---|---|---|---|
| Virginie Grandgirard | Researcher, CEA-IRFM | Numerical Analyst, GYSELA-X main developer | M1-M36 |
| Chantal Passeron | Technician, CEA-IRFM | Support to GYSELA-X development | M1-M36 |
| Julien Bigot | Researcher, CEA-MdlS | Computer Scientist, expert in HPC & I/O | M1-M36 |
| Michel Mehrenberger | Researcher, AMU | Applied Maths., GYSELA-X developer | M1-M36 |
| Postdoc 1 under recruitment | CEA-MdlS | Computer Scientist, PDI | 18 M |
| Postdoc 2 under recruitment | CEA-IRFM | Numerical Physicist, shaping + multi-resolution | 24 M |
| Postdoc 3 under recruitment | CEA-MdlS | Computer Scientist to port the code on ARM-based supercomputers | 7 M |
| Emily Bourne PhD thesis (NUMERICS) | CEA-IRFM / AMU | Computer Scientist, handling complex geometry | M10-M36 |
| Yanick Sarazin | Researcher, CEA-IRFM | Physicist, coordination and reporting | M1-M36 |

*Table 16 – Members of the GYSELA-X team in WP2*

**GYSELA performance results:**

The relative efficiency of GYSELA (weak-scaling starting from 8k cores) is higher than 95% at 65k cores on several supercomputers. GYSELA exhibits an excellent scalability (91% of relative efficiency at 458 752 cores) using all the available computing power of JUQUEEN Blue Gene supercomputer (JSC/IAS, Jülich, Germany), as shown in Figure 9. For a strong scaling experiment, the application obtains a relative efficiency typically larger than 60% at 65k cores (on Curie in 2012). More recently, in 2018, a benchmark of the GYSELA code has been performed on the Irene-Joliot Curie machine (CCRT, France) on the KNL partition. On this machine, the relative efficiency of GYSELA reaches 60% at 32k cores.

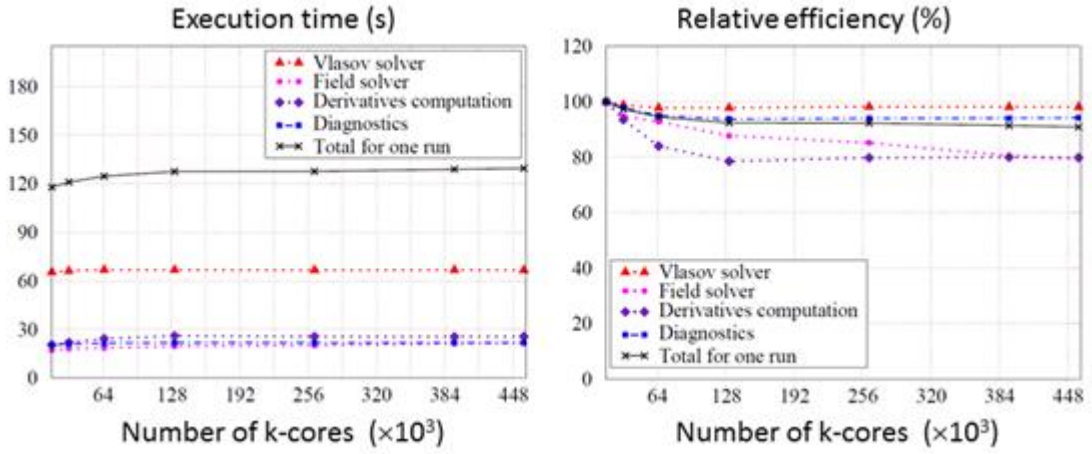


*Figure 9* - CPU time consumption (seconds) and efficiency of parallelization of GYSELA for weak scaling test on Juqueen (May 2013, 4 time steps)

**GYSELA-X updated work plan: detail of subtasks**

In the on-going petascale-exascale transition, GYSELA comes across problems due to the network capability and limited memory performance. As we move forward to extreme parallelism and heterogeneity on modern architectures, the former framework (MPI + OpenMP loops) needs being revisited before deciding which is the best strategy for a modern GYSELA-X code scalable to exascale supercomputers. This is the goal of subtask T2.6.1, where new approaches are tested on reduced codes and models. The outcomes will guide the developments within subtask T2.6.2, which aims at refactoring GYSELA to develop the production code GYSELA-X characterized by groundbreaking physics capabilities and scalable to exascale. A progress report will be written at M18, and a final one at M36.

- T2.6.1: Prototype of GYSELA-X

The long-term effort (started about 1 year ago) to develop a C++ prototype – with minimal set of optimized operators, new data structures, parallelized algorithms and a first OpenMP task approach – has not shown enough advantages with respect to the main expectation. Indeed, it reveals limited efficiency of communication / computation overlap, i.e. less than typically 10%. In addition, it has shown major and crippling drawbacks regarding both readability and maintainability (a paper is submitted on this issue [J. Richard et al., "Fine-grained MPI + OpenMP plasma simulations: communication overlap with dependent tasks", Euro-Par proceedings, to appear in Lecture Notes in Computer Science (2019)]. This cannot fly given the need for physicists to be able to regularly access the code (modify equations, change boundary conditions, etc.). In addition, it would require the entire refactoring (i.e. from scratch) of the code which is no longer a viable option, given the fact that the CEA-IRFM team loses one of the pillar developers of the code, with an expertise on computational science and high-level parallelism which cannot be replaced by non-permanent staff.
Regarding Kokkos implementations, tests have already been performed on C++ prototype applications, in particle-in-cell [V. Artigues et al., "Evaluation of performance portability frameworks for the implementation of particle-in-cell codes", to be submitted (2019)] and semi-Lagrangian schemes [Y. Asahi et al., report in progress (2019)]. The preliminary conclusion is that Kokkos reveals the most adequate framework for performance portability of a parallelized code over a broad spectrum of architectures. These reports will be finalized and submitted to publication. However, since this task requires C++ language which is not the one retained for GYSELA-X, it will not be further developed.
The same holds for some of the initially envisioned subtasks, namely strengthened task implementation and advanced runtime, and communication/computation MPI overlapping.
As stated above, task T2.6.1 has shown that parallelization based on MPI + OpenMP loops is still the best solution, given the constraints of readability and maintainability of a production code like GYSELA-X. In this spirit, the alternative approach using Fortran with enhanced modularity needs being tested and its relevance attested for exascale applications. In this framework, PDI (Parallel Data Interface) likely offers a suitable solution. Indeed, successful results were obtained during EoCoE with respect to checkpoint-restart issues in GYSELA. PDI will be further implemented and its efficiency quantified on prototype versions before being deployed in GYSELA-X if successful. (Julien Bigot and the first postdoc)

- T2.6.2: Advanced GYSELA-X

Four main activities constitute the backbone of the GYSELA-X development, some being backed by the outcomes of task T2.6.1.

*T2.6.2(i): PDI integration + enhanced modularity developments*: A simplified version of PDI has been installed in GYSELA during EoCoE and successfully applied for the writing of restart files. If extended tests reveal efficient, the complete version will be implemented in GYSELA-X and used both for restart files and diagnostics. This effort will be complemented by the restructuring of the code so as to enhance its modularity, with PDI again as the possible solution. The main goal here is to decouple the engine, namely the Vlasov-Poisson solvers, from essential yet peripheral tasks like diagnostics, initialization etc. (Julien Bigot, Virginie Grandgirard, Chantal Passeron and the first postdoc)
A few modules may be written in C/C++ if required. Especially, we plan to develop a C (or C++) routine in the code to implement PoPe [T. Cartier-Michaud et al., Phys. Plasmas 23 (2016) 020702] as a tool for embedded verification and numerical convergence. This is critical to avoid losing CPU time by activating an automatous soft stop if the simulation goes wrong. This issue is particularly exascale relevant. (Virginie Grandgirard and Emily Bourne)

*T2.6.2(ii): Multi-resolution*: the large temperature variation – typically by 2 orders of magnitudes – from the far edge to the very core of tokamak plasmas requires refined meshes. Multi-resolution and/or multi-patch approaches then reveal mandatory to avoid wasting large amounts of CPU time and memory resources. These critical developments require in-depth modifications of the code modules (Virginie Grandgirard, Chantal Passeron, Michel Mehrenberger, Emily Bourne, and the second postdoc)

| | Project Month | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **GyselaX sub-tasks** | **6** | **8** | **10** | **12** | **14** | **16** | **18** | **20** | **22** | **24** | **26** | **28** | **30** | **32** | **34** | **36** |
| T2.6.1: Prototype of GYSELA-X | | | | | | | | | | | | | | | | |
| T2.6.2(i): Advanced GYSELA-X – PDI integration | | | | | | | | | | | | | | | | |
| T2.6.2(ii): Advanced GYSELA-X – multi-resolution | | | | | | | | | | | | | | | | |
| T2.6.2(iii): Advanced GYSELA-X – Handling complex geometry | | | | | | | | | | | | | | | | |
| T2.6.2(iv): Advanced GYSELA-X – Adaptation to ARM | | | | | | | | | | | | | | | | |

*Table 17 – GYSELA-X updated work plan*

*T2.6.2(iii): Handling complex geometry*: This major task requires rewriting most of the operators so as to handle generalized coordinates to be able to address ITER relevant D-shape magnetic geometries. So far, GYSELA can only cope with circular cross-sections. (Virginie Grandgirard, Chantal Passeron, Michel Mehrenberger, Emily Bourne and the second postdoc)

*T2.6.2(iv): Adaptation to ARM-based cluster*: This activity is particularly important since ARM is the possible architecture of future exascale machines in EU and Japan. Therefore, the effort will focus on this type of architecture. Our aim is to reinforce and further develop our collaboration with Japanese teams on this task. Discussions are ongoing, especially with the R-CCS. (Julien Bigot and the third postdoc).

**Risks**

No risk is identified for task T2.6.1. For the substask T2.6.2, risks are identified and envisaged mitigation strategies rely on the possible difficulty and/or delay to hire the three postdocs with adequate expertise:

- For the first and second postdocs, if the work is done at lower speed, we can respectively delay the subtasks T2.6.2(i) and T2.6.2(ii-iii)
- For the postdoc 3, the possibility to perform task T2.6.2(iv) in collaboration with Japanese teams is under consideration.