eFlows4HPC

Enabling dynamic and Intelligent workflows in the future EuroHPC ecosystem

D6.3 Iteration 1 workflows for urgent computing of natural hazards

Version 1.0

Documentation Information

Contract Number	9555558
Project Website	www.eFlows4HPC.eu
Contratual Deadline	31.08.2022
Dissemination Level	PU
Nature	Other
Author	Carlos Sánchez Linares (UMA)
Contributors	Jorge Macías (UMA), Marc de la Asunción (UMA), Marisol Monterrubio (BSC), Juan E. Rodriguez (BSC), Steven J. Gibbons (NGI), Marta Pienkowska (ETH), Jacopo Selva (INGV), Louise Cordrie (INGV).
Reviewer	Yolanda Becerra (BSC)
Keywords	Urgent computing, workflow implementation

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955558. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Germany, France, Italy, Poland, Switzerland, Norway.

Change Log

Version	Description Change
V0.1	Proposed ToC
V0.2	First draft
V0.3	Submitted for revision
V0.4	Revised based on reviewer's comments and suggestions
V1.0	Final version with reviewer's acceptance.

Table of Contents

1. Executive Summary

This deliverable describes the status of the implementation of the two workflows corresponding to *Pillar III: Urgent computing for natural hazards*. With this implementation, it is intended to use the eFlows4HPC components to boost urgent computing simulations for earthquakes and tsunamis.

With this first implementation, and after a deep analysis of both workflows dependencies, it has been possible to establish a first iteration that contemplates the characteristics expected at this point in the project, having ported most components of UCIS4EQ and PTF/FTRT workflows to PyCOMPSs , adding new functionalities such as a near-real-time source-estimation manager to initialize PTF ensemble forecast based on earthquake data, and also performing numerous simulations using HPC resources as a testing phase.

This development, together with what is expected in the following phases, will allow faster end-to-end runs to be executed, with more robust and reliable workflows, and outcomes more usable to potential end-users, thus achieving the main objective of WP6 in this project.

2. Introduction

Earthquakes and consequent tsunamis are unpredictable events and capable of catastrophic impact on human lives, infrastructure, and economy. The unpredictability of their occurrence poses a challenge to the scientific community, as an assessment of the impact severity needs to happen in a very limited time and based upon relatively sparse data. Accurate, efficient, and rapid mathematical/computational modelling is thus called upon to provide hazard assessments.

The workflows to be implemented need to provide timely, accurate, and reliable information to decision makers. Both earthquake and tsunami workflows involve rapid simulation of the hazard phenomena and these parts of the processes invariably need to be performed in HPC infrastructure. The workflows require initialization immediately following an event, urgent access to HPC facilities, and post-processing and visualization steps which make their output available to the end users.

The main objective of Pillar III is the development and adaptation of workflows, both for earthquakes and tsunamis, providing a common orchestration to easily integrate into HPC environments.

This deliverable accompanies the first versions of the workflows developed in Tasks 6.3 and 6.4: iteration 1 for “workflows for urgent computing of natural hazards”, which are widely detailed in Deliverable 6.1 (see URL in reference section).

3. Implementation status of Pillar III workflows

This section describes in detail the current implementation status of the two workflows: the seismic workflow (UCIS4EQ) and the tsunami workflow (PTF/FTRT). Both workflows of Pillar III have been developed independently, making use of different HPC simulation codes and with different sets of requirements, as well as pre- and post-processing strategies and approaches.

3.1. The Seismic Workflow (UCIS4EQ)

The Urgent Computing Integrated Services for Earthquakes (UCIS4EQ) workflow is a suite of microservices defined as building blocks i.e., independent components of the system that carry out specific tasks. UCIS4EQ has the potential to deliver more accurate short-time reports of the consequences of moderate to large earthquakes. UCIS4EQ rapidly provides synthetic estimates of ground motion parameters, such as peak ground velocity, peak ground acceleration, or shaking duration, with very high spatial resolution. Such outcomes can be used to analyse the overall ground motions in the area as well as potential impacts on key infrastructures that could produce collateral risks (fires, dam rupture, among others).

The UCIS4EQ implementation in the eFlows4HPC software stack in the Iteration 1- Phase 2 (M7-M20) has been defined by five main activities described below.

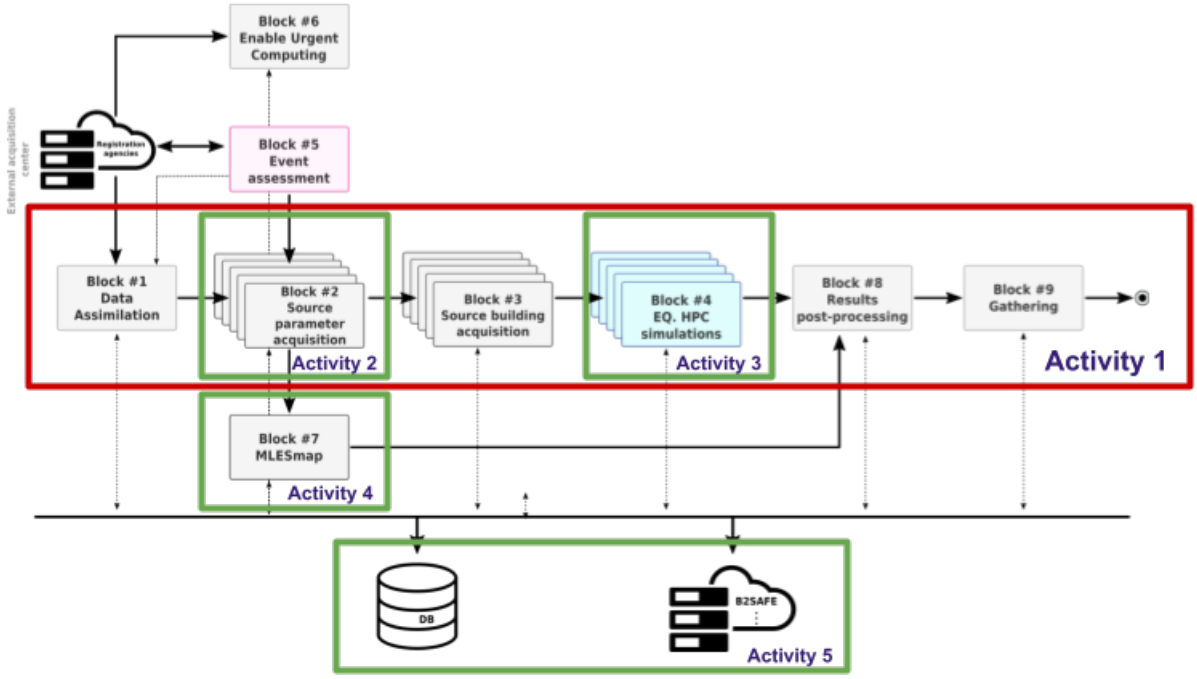

Figure 1: A simplified representation of the UCIS4EQ workflow with the eFlows4HPC activities described in this deliverable marked with red and green rectangles.

UCIS4EQ Activity 1: Porting UCIS4EQ to PyCOMPSs for a new release of the application (BSC).

At the beginning of the first phase of the project, we identified the support in PyCOMPSs (Tejedor et al., 2017) for the micro-services design structure of the UCIS4EQ service as a key requirement. The requirement has been satisfied and PyCOMPSs has been integrated as a WF manager to orchestrate most of the building blocks of the UCIS4EQ shown in the red rectangle of Figure 1. PyCOMPSs thus replaces the existing WF manager emulator that has been managing the tasks sequentially in the previous UCIS4EQ version. Given that the code licensing is still pending and remains an ongoing task, this first migration of PyCOMPSs is presented in this first release via a mock-up version of UCIS4EQ that mirrors the actual implementation in the full service and that allows to analyse the dependencies between inputs and outputs of the different microservices. It should be highlighted that the UCIS4EQ implementation is the first service where PyCOMPSs is orchestrating microservices and not only tasks. We thus successfully demonstrate new capacities of PYCOMPSs - developed during the eFlows4HPC project - as a Workflow manager.

The PyCOMPSs implementation with the restructured version of the service (see Activity 5 below) has been tested on CSCS' Piz Daint with a full high-frequency end-to-end execution on GPUs, spanning 22 HPC jobs on 90 GPUs each with a wall-time of about 80 minutes per job.

See Section 4 to the link of the mock-up version

UCIS4EQ Activity 2: Managing the initialization and the updates of ensembles of seismic sources for uncertainty quantification within UCIS4EQ, as a prototype (INGV).

In this activity we were working on implementing a new method to manage the uncertainty on the seismic source parameters through the definition of an ensemble of simulations that explore such uncertainty. The implemented method, named SeisEnsMan, is derived from the method described in Selva et al. (2021) for the Probabilistic Tsunami Forecasting (PTF). This component shares commonalities with the equivalent activity (Step 1) in the tsunami workflow described in Section 3.2, harmonizing the workflows for earthquakes and tsunamis generated by the same seismic source.

The SeisEnsMan block managing the ensemble in UCIS4EQ is derived from Step1 of the tsunami workflow, and it introduces several specializations required for better managing the specific needs of seismic sources. In particular, a finer sampling of source parametrization is foreseen to manage near-source uncertainty, which is essential for seismic applications that are focused in target areas closer to the seismic source than the ones for tsunamis.

The new parametrization has been tested by studying the convergence of the uncertainty quantification, adopting standard Ground Motion Prediction Equations (GMPE) models to evaluate the potential seismic impact of the different sources. The defined analysis will be the basis for testing the convergence performance of the ensembles considering a set of past earthquakes, as well as for monitoring the need of further simulations in real-time applications of UCIS4EQ.

In the present version, SeisEnsMan takes as input the essential seismic source parameters (magnitude and hypocentral location) and provides an ensemble of scenarios describing the source uncertainty and a list of probabilities describing how applicable each scenario is to the target event. In the next iteration, we anticipate an update to SeisEnsMan to allow an update of the list of scenarios and their probabilities considering new information (for example new source information, e.g. focal mechanisms, or updated seismic observations, e.g. from accelerograms), and through developing a tool to monitor the convergence of uncertainty quantification based on GMPE models.

The most recent version of SeisEnsMan is available in the the GitLab repository listed in Section 4.

SeisEnsMan has recently been embedded into a Docker container to fulfill the requirements of integration within the UCIS4EQ micro-services design structure (for details on the UCIS4EQ design, see Deliverable 6.1). Like other elements of the UCIS4EQ, the Docker container is now built as an application that is ready to be called within the workflow, but will be run with its own specific environment. Integration and testing of the SeisEnsMan component in the UCIS4EQ service is now ongoing.

UCIS4EQ Activity 3: Producing new Earth models for the regions of interest defined in Task 6.2 (ETH).

In Iteration 1 Phase 2, we integrated generating models from the Collaborative Seismic Earth Model (CSEM, see URL in reference section), developed at ETH Zurich, into the Salvus_urgent_wrapper package that integrates the pre- and post-processing for the Salvus wave propagation software into UCIS4EQ. A model for any given region can now be extracted and interpolated directly onto a computational mesh, without the intermediate step of generating a separate model file for the region (thus reducing the number of interpolation steps). The CSEM model is a global multi-scale model, giving UCIS4EQ the flexibility of defining a new computational domain anywhere: effectively we now have a global database of available Earth models within the service. The UCIS4EQ Salvus_urgent_wrapper package that provides the Salvus pre- and

post-processing can either generate meshes with CSEM on-the-fly (that is, right before the simulation), or they can be pre-computed and subsequently read-in for pre-processing.

Meshes with model values from CSEM interpolated onto the computational grid have been generated for the regions and events defined in Deliverable D6.2 (see URL in reference section) for maximum frequencies of 2Hz and 5Hz. Although the global CSEM model is not event-specific, the meshes in the current UCIS4EQ implementation are event-specific given the computational resources and the status of the service development, and thus include very small sub-models.

The six meshes (three events, two meshes each) reside on the two clusters where we run the UCIS4EQ service, that is on MareNostrum4 (BSC, Spain) and on Piz Daint (CSCS, Switzerland):

1. The Mediterranean Sea: the 2017 M6.6 Kos-Bodrum earthquake, 120 km x 100 km domain @2Hz and @5Hz (385 MB and 2.7 GB, respectively).
2. Mexico: the 2017 M7.1 Puebla earthquake, 175 km x 210 km domain @2Hz and @5Hz (1.3 GB and 12 GB, respectively).
3. Iceland: the 2000 (June 21) M6.5 SISZ earthquake, 135 km x 85 km domain @2Hz and @5Hz (645 MB and 4.7 GB, respectively).

In the next phase, we will attempt to refine selected regional models with short-period full-waveform inversion (FWI), provided that the available seismic data allows for further reductions in the misfit.

UCIS4EQ Activity 4: Developing ML methodology as proof of concept using dislib and EDDL libraries (BSC).

In this activity, we have been developing the proof-of-concept to enable ML-based methodologies that can complement or replace the 3D physical simulators and hence significantly reduce the time-to-solution of our urgent computing solution and/or help us explore uncertainties quickly and reliably. To reach this objective, the MLESmap building block is included in the UCIS4EQ workflow as a method set up before an operational execution of UCIS4EQ. MLESmap aims to quickly assess ground-motion intensity maps through ML inferences trained from physics-based earthquake simulations. In order to feed our MLESmap technology, a large set of intensity measures for likely earthquakes in the study region is generated, which can be obtained from 3D scenario simulations. We are familiar with the Cybershake software, which is installed at MareNostrum4 in collaboration with the Southern California Earthquake Center (SCEC). The Cybershake package (Graves et al., 2011) has reciprocity capabilities, which allows the modelling of an almost arbitrary number of earthquakes for a given area, scaling the cost linearly by the number of spatial sites at which we need to record seismic intensities. Once the dataset is generated, we can feed out MLESmap technology and obtain fast analogues to the scenario simulations, which may render obsolete previous efforts toward real-time shake estimates. Early tests of the technology (Monterrubio-Velasco et al., 2021) show promising potential for our MLESmap, keeping substantial accuracy when compared to a subset of validation scenarios. Our approach (i.e., simulate, train, deploy) can result in the next generation of shakemap estimates, capturing physical information from wave propagation (directivity, topography) at the velocity of simple empirical ground motion prediction equations. Moreover, as our synthetic catalogues will never contain all possible future events, we aim at having models capable of successfully interpolating non-trained events accurately.

For a proof of concept for this activity we have used one of the largest and most validated synthetic catalogues in collaboration with the SCEC, that is the CyberShake 15_4 Study (for details see URL in reference section), developed for the Los Angeles region in Southern California. A large number of synthetic events (~700 million hypothetical earthquakes) make this catalogue an ideal database with which to develop the MLESmap technology.

In our particular application, the problem to be solved has been identified as a regression formulation where the objective is to infer the ground motion proxies measured in Pseudo Spectral ground Acceleration (PSA) for different periods at different sites or locations over the subsurface. To reach the objective we

select the Random Forest (RF) (Cutler et al., 2012) regressor integrated in the dislib (Álvarez Cid-Fuentes et al., 2019) and the Neural Network (NN) through the EDDL (see URL in reference section) package.

The RF regressor was implemented and optimized in the dislib package based on the needs of the MLESmap in the first half of this project. During the first interaction of Task 6.3, continuing work with the dislib developers has been done to improve the capacities of the algorithm in order to optimize and provide the expected results. In particular, the inverse transform scaler method was implemented to solve the need to convert the ML inferences into the appropriate physical units after training. Also, an optimization of the block size was performed in the RF regressor to improve the parallelizing in the algorithm due to the large dataset. Moreover, to execute the RF hyperparameter searching through the Grid Search method, a splitting of the parameters in different executions using a total of 16 nodes of the MareNostrum4 was done to improve the time-to-solution due to the large dataset.

Regarding the NN algorithm also chosen to solve this application, an effort has been made to determine a suitable network topology for the selected dataset. In order to achieve this, we started with a collection of basic multilayer perceptrons (MLPs), with different numbers of layers and neurons per layer. Several techniques have also been tested such as 1) regularization, data normalization, and batch normalization in order to tackle the problem of non-generalization of neural networks; 2) dropout and different activation functions, for the problem of vanishing and exploding gradients; and, finally, 3) different learning rate schedulers to deal with the local minima deadlock optimization problem. After carrying out different tests with a reduced data set, the best neural network topology was determined to be an MLP with 4 hidden layers, and 64, 128, 128, and 64 neurons per layer, respectively; post-batch-normalization (after the activation function); a softplus activation function for the hidden layers; and an Annealing Warm learning rate scheduler.

The proof of concept of the MLESmap developed in this activity including the network topology and the best hyperparameters for RF and NN algorithms will be applied in the second phase of this project for the Iceland region proposed as Use Case in the Deliverable 6.2 (see URL in reference section).

UCIS4EQ Activity 5: Re-structuring of data management of the UCIS4EQ service (BSC, ETH).

In order to facilitate the implementation of new regions in UCIS4EQ, the code required significant restructuring. As the concept of the workings of the service changed with development, many of the tools were run directly on the cluster and with the cluster serving as a data repository - as opposed to the initial idea of only the HPC-intensive jobs running on the cluster. The data transfers to and from the cluster have proven to be too time consuming for high-resolution runs and therefore many services are sent to the HPC cluster, with some data residing directly there. This adaptation, however, resulted in a convoluted structure of implementing new test cases.

UCIS4EQ, therefore, has been restructured to accommodate for the HPC and for the B2DROP data repository in a coherent way, allowing to implement new test cases with ease and facilitating service deployment. The main changes include:

- Collapsing the definition of domains and regions to a single regional entity. This is with mesh-masking in mind for future developments (likely beyond the eFlows4HPC project), where the mesh will be for a large region and the service will mask out most of the mesh, leaving only a small mesh around the given event.
- Including folders in the StaticDataMapping, which can now link to both specific files and to folders. The UCIS4EQ is also able to check the content of the remote folders for each region for correct files.
- Application-defined folder pattern, with the Salvus-appropriate folder pattern currently available.
- The definition of a specific execution policy for the maximum frequency (e.g. highest available) and source ensemble method selection.
- The restructuring with the per-use-case coherent data structure forms a basis for the second phase of the eFlows4HPC project, where we expect to use the eFlows4HPC tools to facilitate the

deployment of the service that includes an automatic data upload to the HPC cluster. Further requirements which necessitate a thorough evaluation within the context of the eFLows4HPC software stack and a subsequent definition of the underlying mechanism include an automatic data sync with the main repository, as well as a final file-check at runtime that makes sure that all files are available (or sync and upload for the main repo if not available).

Summary of T6.3 for UCIS4EQ

In the table below we summarize the above activities and highlight the status of the WF and of the different components prior to the project and reached at the first iteration.

Table 1. Status of the earthquake WF first iteration and components

Workflow	**Status before project**	**Phase I status**
UCIS4EQ (Activity 1 and Activity 5)	No integrated WF manager; only a sequential WF manager emulator. Difficult integration of new regions.	PyCOMPSs WF manager integrated in the full service and a mock-up version released that demonstrates the dependencies between inputs and outputs of the different microservices. Restructuring of the data management, which allows for easier use-case implementations and forms a basis for further work in phase 2 of the project on automatic deployment.
Components / Building Blocks	**Initial status**	**Phase I status**
ML / AI (Activity 4)	None	Proof of concept developed as a stand-alone tool.
SeisEnsMan (Activity 2)	Prototype used in the Tsunamis workflow.	Adaptation of the SeisEnsMan for earthquakes and porting to Docker for integration with UCIS4EQ.
Earth Models (Activity 3)	None	Integration of the global Collaborative Seismic Earth Model within the Salvus service of UCIS4EQ, enabling velocity model extraction for any desired region directly on the computational mesh.

Next Steps of T6.3 for UCIS4EQ

Activity 1 and 5: We are going to focus on data management through the Data Logistic Services, streaming data source, HPC job management through PyCOMPSs, workflow malleability, and execution robustness. Moreover, in the 2nd iteration the UCIS4EQ requirements of the eFlows4HPC software stack will be revised in order to account for the ongoing evolution of the service.

Activity 2: In the 2nd iteration we need to develop a prototype for Uncertainty Quantification incorporated in Block 8 (see Fig. 1) from the source ensemble developed in the 1st iteration.

Activity 3: We will attempt to refine selected regional models with short-period full-waveform inversion (FWI), provided that the available seismic data allows for further reductions in the misfit. We will then assess the impact of such model refinements on the ground shaking synthetics.

Activity 4: We aim to set up the MLESmap methodology for the South Iceland Region. The activity can be subdivided into two stages: first, the generation of a database, and second, the ML training is based on the DA/ML libraries of the eFlows4HPC software stack optimized for execution in HPC environments. The models will be integrated in the UCIS4EQ workflow as part of Block 7 (see Fig. 1).

3.2. The Tsunami Workflow (PTF/FTRT)

The Tsunami Workflow (PTF – Probabilistic Tsunami Forecast/FTRT - Faster Than Real Time) seeks to provide a forecast of tsunami impact following a large offshore or near-shore earthquake. The uncertainty in the source is dealt with by considering a (potentially very large) ensemble of earthquake scenarios. For each such scenario, an efficient numerical simulation needs to be performed in which the impact on the coastlines of interest is calculated. Just as numerical weather prediction generates a probabilistic forecast based on the outputs from multiple ensemble members, PTF calculates a probabilistic prediction of tsunami impact based upon the outputs of the individual simulations and their scenario probabilities. The complete workflow is displayed in Figure 2.

For Iteration 1- Phase 2 (M7-M20), we simplified the overall workflow foreseen in Deliverable 6.1 by defining 5 sequential tasks, hereinafter called STEPs, as reported in Figure 2. This simplified workflow includes:

- Step1: definition of the ensemble of scenarios to be modelled, based on seismic source uncertainty
- Step2: simulation of tsunami sources in the ensemble
- Step3: post-processing of single scenario output and production of PTF output files
- Step4/5: aggregation of the results and visualization

In the following, we describe the single components.

- **STEP 1 – Ensemble initialization**

In STEP 1, we initialize in real-time the ensemble that manages the uncertainty on the seismic source for the Probabilistic Tsunami Forecasting (PTF). This typically happens few minutes after the earthquake, when the initial magnitude and hypocentral location become available. The ensemble may potentially be updated with time, as new information becomes available. The methodology behind the ensemble definition is described in Selva et al. (2021). STEP 1 takes as input the main earthquake parameters and provides as output a list of scenarios to be simulated in STEP 2 and a list of probabilities to be applied in the aggregation of the PTF, in STEPs 4/5. Both files are txt ASCII files.

The starting point for the STEP 1 code was the Matlab implementation developed in Selva et al. (2021) and available at https://github.com/INGV/matPTF. From this code, a version isolating STEP 1 has been derived and included in the PTF workflow developed in the ChEESE project (see URL in References section). Both versions are entirely in MATLAB and cannot be run in an HPC server. Thus, they have always been run in a

separate server, and the STEP 1 output transferred to a HPC server for execution of the simulations in STEP 2.

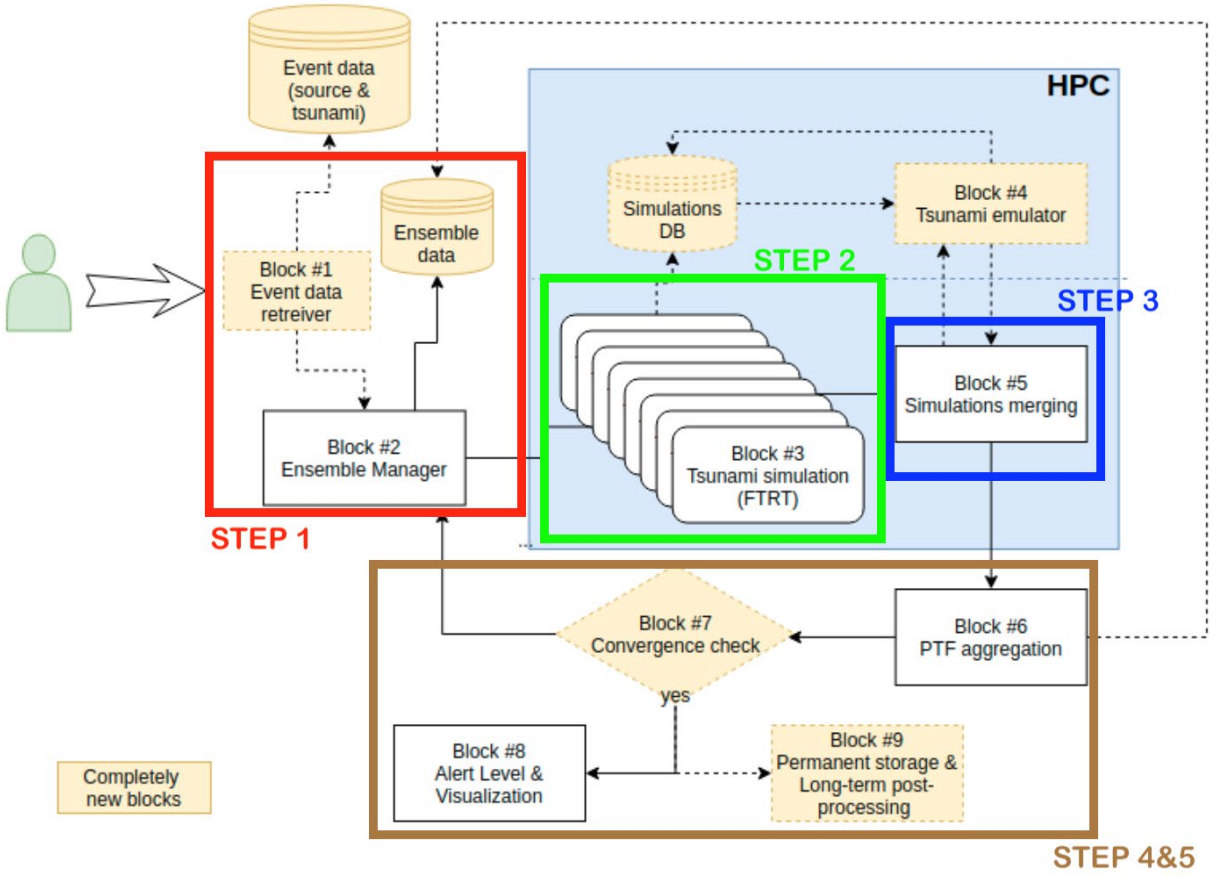

Figure 2: PTF/FTRT workflow implementation for Iteration 1- Phase 2 compared with its final implementation foreseen for Iteration 2, as described in Deliverable D6.1.

These original versions were also entirely based on the method described in Selva et al. (2021). This method defines pretty large ensembles, with a number of members that ranges from tens to hundreds of thousands of members. This allows a simplification of as much as possible of all the remaining parts of the workflow, to decrease as much as possible the computational effort of individual simulations.

In Iteration 1- Phase 2, we ported the original codes to Python, for portability and so that STEP 1 can be easily run on the same server on which the subsequent steps are run. The initialization of the ensemble depends only on small files defining the main earthquake characteristics. This enables an updating of the ensemble when new information becomes available, as the code may be rerun locally simply updating a few input parameters, providing an updated set of output files (scenario list and probabilities), enabling a new aggregation at STEPs 3/4. This will become fundamental in Iteration 2 – Phase 3, when the updating of the ensemble will be fully implemented in the workflow.

This new version of STEP 1 in Python adds also a new management of the uncertainty on the source parameters for the definition of the ensemble, enabling full control on the size of the ensemble and a significant reduction of the number of required simulations to obtain a satisfactory description of source uncertainty.

Control of the ensemble size is fundamental to better planning and management of the use of resources within the WF. This was only partially possible in the previous versions of STEP 1 (though the parameter sigma: see Selva et al. 2021) and, in these versions, the exact number of simulations was not directly controlled, nor could be reduced below a given threshold. Now, the number of simulations is required in input.

A new approach toward the definition of the ensemble members has been adopted to obtain a more efficient quantification of the uncertainty based on a smaller number of ensemble members. The procedure has been tested for several past earthquakes using the precomputed simulations adopted in Selva et al. (2021) and derived from the NEAMTHM18 hazard model (Basili et al. 2018, 2021). The selected past events included two of the case studies adopted in eFlows4HPC (Deliverable D6.2), that is, the 2003 Mw6.8 Zemmouri-Boumerdes and the 2017 Mw 6.6 Kos-Bodrum earthquakes. The ability to converge faster towards the full description of the source uncertainty with a smaller number of simulations will also enable a modification of the subsequent steps, allowing production of a PTF with a smaller number of resource-intensive simulations.

In next iteration, we anticipate an update to STEP 1 by allowing revision of the list of scenarios and their probabilities considering new information, like for example new source information (e.g., focal mechanisms) or tsunami observations (e.g. accelerograms), and by developing a tool monitoring the level of convergence of uncertainty quantification. These new functionalities are already in development.

STEP 1 call has been finally embedded in PyCOMPSs, to integrate this step into the overall workflow managed through PyCOMPSs. STEP 1, as a non-time-consuming part of the workflow, is called as one single task providing the files required for STEP 2.

- **STEP 2 – Tsunami simulation**

STEP 2, in charge of performing the simulations using Tsunami-HySEA Monte Carlo version (Macías et al, 2019 & Escalante et al, 2017), has been modified in order to take advantage of the orchestration provided by PyCOMPSs. Once STEP 1 provides the ensemble file with the scenarios to be simulated, the first task is carried out: grouping the scenarios into different chunks. This is one of the substantial improvements with respect to the previous state of the workflow, since with the implementation in PyCOMPSs, this task is carried out in parallel for each one of the chunks, instead of being processed sequentially.

For each of these chunks, the execution task of the Tsunami-HySEA simulator is established. This one is called through a COMPSs MPI invocation, allowing the user to set the constraints for each run, such as the number of cores required explicitly or indicate that the value of a constraint is specified on an environment variable.

With this new implementation of the core of STEP 2, it is possible to launch a single job to the queue system that implicitly carries out the parallel execution of as many simulations as the user indicates at the beginning of the WF, dividing into internal jobs, and allowing the traceability of each processes involved to be observed.

- **STEP 3 – Simulation post-processing**

STEP 3 of the workflow exploits the features provided by the Ophidia framework (see URL in References section) to compute the following operations for each of the time series output by the tsunami simulations. These relate to the wave amplitude variable: the maximum, the minimum, the peak-to-trough, the green's amplification and the removal of the offset with respect the sea level before the tsunami, if needed.

The starting point for the STEP 3 code consisted of two Python scripts developed in the project ChEESE. The first script read the individual simulation results (.nc or NetCDF files), extracting the relevant information and performing several other operations, and saving the extracted information in new .nc files. The second script read all the individual extracted .nc files to produce a single .nc file to be given in input to the PTF aggregation of STEP 4/5.

The use of the Ophidia framework, due to its capabilities to manage and manipulate NetCDF files using an in-memory approach, represents an improvement with respect to the first python-based version which instead required continuous I/O operations from disk to save and then retrieve the outputs for the final merging phase.

In more detail, starting from a large number of files (one for each tsunami simulation), some postprocessing operations are performed in-memory on groups of files and a single file for each computed variable is saved on disk (for a total of 8 files for each group) before a final merge of all the results belonging to the group. Figure 4 provides a schematic representation of the workflow defined by this step.

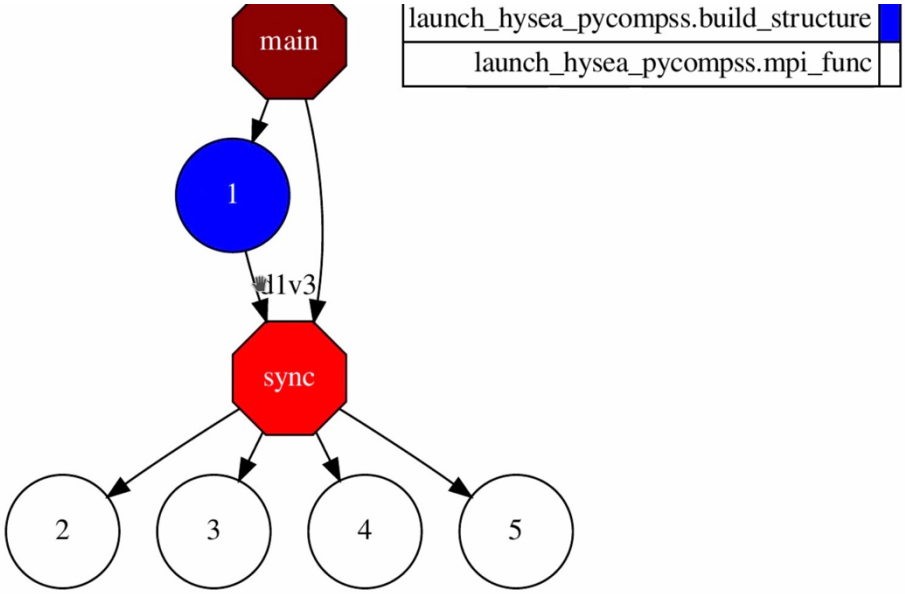

Figure 3: Example of graph for the STEP2 minimal workflow with an ensemble of 8 scenarios using n=2 as the number of scenarios per chunk.

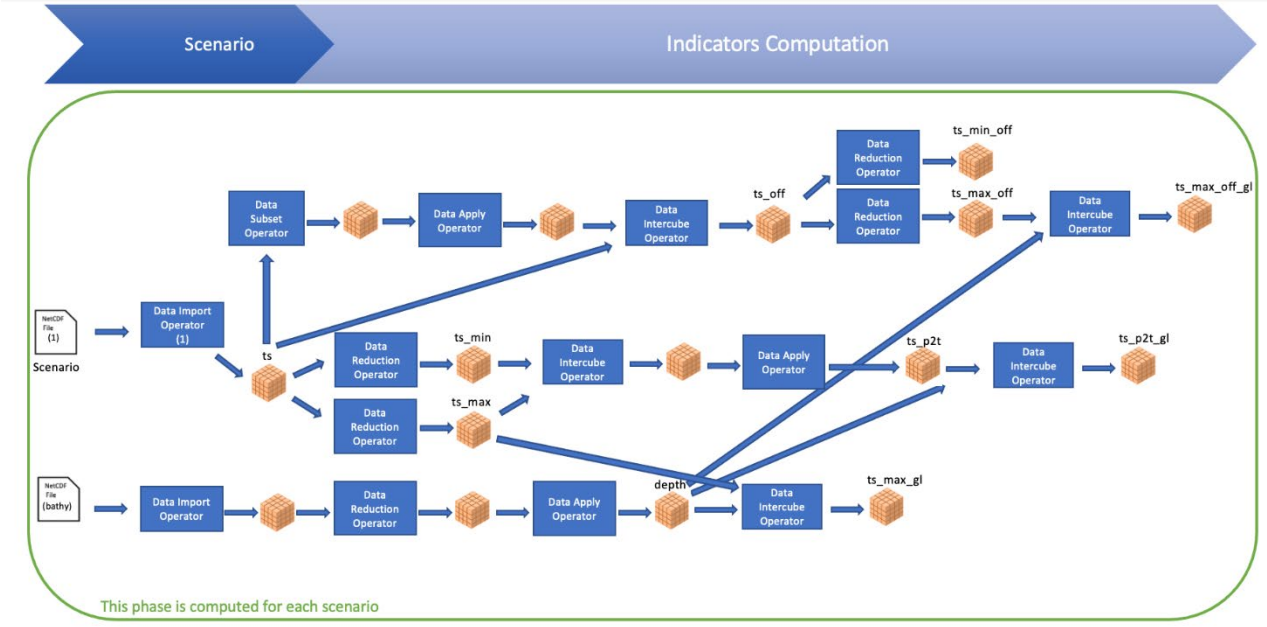

Figure 4: Internal steps for the HPDA-based indicators computation block

Another improvement is the workflow orchestration by means of PyCOMPSs exploiting parallel execution. In Figure 5 is shown the PyCOMPSs tasks graph generated at the end of the workflow. For the sake of clarity, the graph shows the execution of the workflow for a single group of two scenarios; in case of multiple groups the whole graph will be repeated.

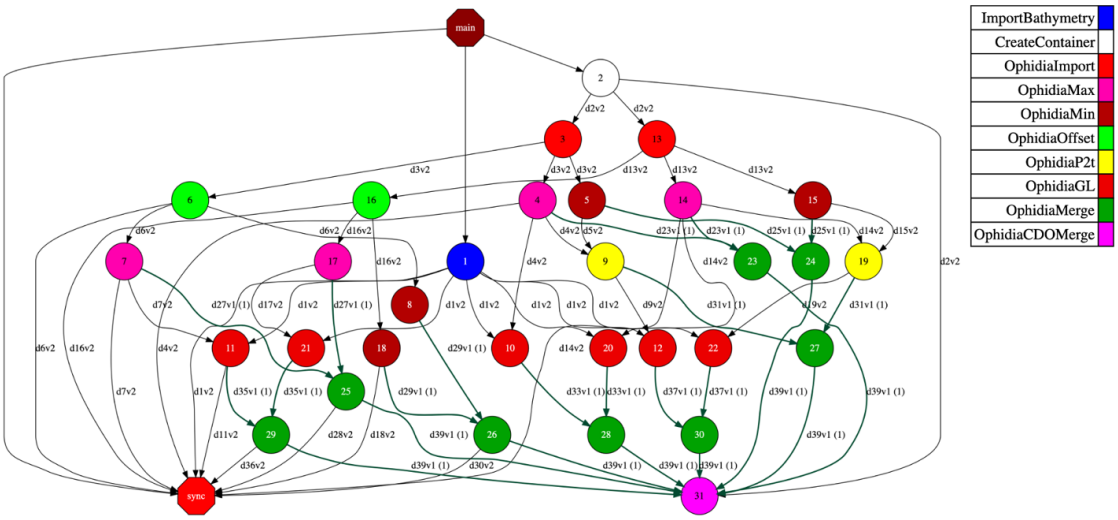

Figure 5: PyCOMPSs task-dependency graph for the STEP3 workflow

The code showed at Figure 6 contains the declaration of tasks in PyCOMPSs and the invocation of them on 864 files (in groups of 72 scenarios).

The first task computes the import into Ophidia of the bathymetry variable that is used in the subsequent task related to the Green Law formula.

The *OphidiaImport()* task performs the import into Ophidia of the Wave amplitude related to a scenario, in order to: (i) compute the maximum and minimum with and without offset using the reduce operator, (ii) the peak to trough (i.e., the difference between maximum and minimum divided by two) using the *intercube* operator and the *mul_scalar* primitive and (iii) then the indices related to the Green Law. All these operations are computed in the other tasks.

The *OphidiaMerge()* task regards the merge of the previous created datacubes adding the scenarios dimension for each calculated variable and exporting the result as a NetCDF file, while the last task puts all the variables together in the same file to obtain the result file.

```
#Declaring tasks
@task(returns=object)
def importBathymetry():
    bath    =    cube.Cube.importnc2(src_path=    'bathymetry.nc',    measure='deformed_bathy',
imp_dim='time', exp_dim='grid_npoints')
    bath_reduce=bath.reduce(operation="max")
    depth            =            bath_reduce.apply(query="oph_math(oph_math(oph_predicate('x-
1','<0','1','x'),'SQRT'),'SQRT')")
    return depth
@task(returns=object)
def OphidiaImport(scenario, container):
    ts    =    cube.Cube.importnc2(src_path=    filePath,    measure='eta',    imp_dim='time',
exp_dim='grid_npoints')
    return ts
@task(returns=str)
```

```
def OphidiaMax(ts):
    ts_max = ts.reduce(operation="max")
    return ts_max.pid
@task(returns=str)
def OphidiaMin(ts):
    ts_min = ts.reduce(operation="min")
    return ts_min.pid
@task(returns=object)
def OphidiaOffset(ts):
    firstRow=ts.subset(subset_dims="time",subset_filter="1",subset_type="index")
    ts0 = firstRow.apply(query="oph_extend('OPH_FLOAT','OPH_FLOAT',measure,961)")
    ts_off = ts.intercube(cube2=ts0.pid, operation='sub')
    return ts_off
@task(returns=str)
def OphidiaP2t(ts_max_pid, ts_min_pid):
    diff = ts_max.intercube(cube2=ts_min_pid,operation="sub")
    ts_p2t = diff.apply(query="oph_mul_scalar(measure,0.5)")
    return ts_p2t.pid
@task(returns=str)
def OphidiaGL(cube_pid, depth):
    gl_cube = ts_cube.intercube(cube2=depth.pid,operation="mul")
    return gl_cube.pid
@task(returns=object, cubes=COLLECTION_IN)
def OphidiaMerge(cubes, name, group):
    pids = '|'.join(cubes)
    ts_merge = cube.Cube.mergecubes2(cubes=pids, dim="scenarios")
    ts_merge.exportnc2(output_name=name+str(group))
cube.Cube.script(script='rename_variables_group.sh',args=name+'|'+str(group)
    return ts_merge
@task(dependencies=COLLECTION_IN)
def OphidiaCDOMerge(group, scenarios, container, dependencies):
    cube.Cube.script(script='merge_files_group.sh', args=str(group)+'|'+str(scenarios))

#Invoking tasks
depth = importBathymetry()
group=0
scenarios=2
for j in range(0,864,72):
    group+=1
    for i in range(scenarios):
        #Import of all scenarios files
        datacubes[i] = OphidiaImport(str(i+j+1).zfill(3), container)
        ts_max_cubes[i] = OphidiaMax(datacubes[i])
        ts_min_cubes[i] = OphidiaMin(datacubes[i])
        ts_off_cubes[i] = OphidiaOffset(datacubes[i])
        ts_max_off_cubes[i] = OphidiaMax(ts_off_cubes[i])
        ts_min_off_cubes[i] = OphidiaMin(ts_off_cubes[i])
        ts_p2t_cubes[i] = OphidiaP2t(ts_max_cubes[i], ts_min_cubes[i])
        ts_max_gl_cubes[i] = OphidiaGL(ts_max_cubes[i], depth)
        ts_max_off_gl_cubes[i] = OphidiaGL(ts_max_off_cubes[i], depth)
        ts_p2t_gl_cubes[i] = OphidiaGL(ts_p2t_cubes[i], depth)
    #Merge all scenarios
    ts_max_cube = OphidiaMerge(ts_max_cubes, "max", group)
    ts_min_cube = OphidiaMerge(ts_min_cubes, "min", group)
    ts_max_off_cube = OphidiaMerge(ts_max_off_cubes, "max_off", group)
    ts_min_off_cube = OphidiaMerge(ts_min_off_cubes, "min_off", group)
    ts_p2t_cube = OphidiaMerge(ts_p2t_cubes, "p2t", group)
    ts_max_gl_cube = OphidiaMerge(ts_max_gl_cubes, "max_gl", group)
    ts_max_off_gl_cube = OphidiaMerge(ts_max_off_gl_cubes, "max_off_gl", group)
    ts_p2t_gl_cube = OphidiaMerge(ts_p2t_gl_cubes, "p2t_gl", group)
    #Merge all files in a single output file
    OphidiaCDOMerge(group, scenarios, container, [ts_max_cube, ts_min_cube, ts_max_off_cube,
ts_min_off_cube, ts_p2t_cube, ts_max_gl_cube, ts_max_off_gl_cube, ts_p2t_gl_cube])
```

Figure 6: Ophidia tasks declaration and invocations

- **STEPs 4/5 – Aggregation and Visualization**

In STEPs 4/5, the Probabilistic Tsunami Forecasting (PTF) is produced and visualized by aggregating the results of individual simulations produced in STEPs 3 and 4, and the probabilities produced in STEP 1.

The starting point for STEPs 4/5 consisted of MATLAB codes developed in Selva et al. (2021) and available at Section 4 as the name matPTF. From this code, a version isolating STEPs 3/4 has been derived and included in the PTF workflow developed in the project ChEESE. Both versions are entirely in MATLAB and cannot be run in a HPC server. Thus, they have been always run in a separate server, manually transferring the output of STEPs 2 and 3 from the HPC server to a local server.

In Iteration 1- Phase 2, we ported the original codes to Python, so that also these steps can be easily run in the same server in which all the previous steps are run. This implies that the main PTF results will not depend of data transfer, allowing a better integration and organization of all output files. In this phase, a minimum visualization has been developed, while more advanced tools are foreseen in next iteration.

- **ML / AI: Tsunami Modelling Emulation**

Several ML use cases have been explored in order to predict results of tsunami simulations. Specifically, we have predicted different alert levels (a classification problem), and maximum water heights and arrival times (a regression problem). In both cases, multilayer perceptron (MLP) networks are used, and the input of the network are the nine Okada parameters (longitude, latitude, depth, length, width, strike, dip, rake, and slip) of the earthquake that causes the tsunami. An early stopping strategy is employed to prevent the model from adjusting too much to the training sets.

For the classification problem, we have considered tsunamis in the GC06 region of the Gulf of Cadiz, as defined in Matias et al. (2013), where the nine Okada parameters vary within the ranges of plausible values for this region, according to Matias et al. Four alert levels have been considered depending on the maximum water height: green (<1 cm), yellow (>1 cm and <30 cm), orange (>30 cm and <1 m), and red (>1 m). The finest spatial resolution used is 40 m. We have trained two MLP networks for predicting the alert level in the Spanish cities of Cadiz and Rota, respectively. A total of 110,000 samples have been obtained using Tsunami-HySEA.

We have obtained precision and recall values between 91.6 and 96,1 % for all the alert levels in the test sets using the best model, with an average value greater than 93 %. Using ensembles of the three best models for each city we have been able to improve the results between 92.4 and 96.7 %, with an average value greater than 94 %. The inference time of these ensembles is very fast, lasting approximately 150 ms in a Tesla A100 to infer the results of 2000 samples.

For the regression problem, we have taken as a reference the Caribbean 2013 LANTEX scenario (see the LANTEX 13 reference), where the Okada parameters vary around the values associated with this scenario. Two MLP networks have been used to train two models to predict the maximum water height and the arrival times respectively of the tsunamis, at 6 points located near the following coastal cities of Puerto Rico: Mayagüez, Guánica, Ponce, Salinas, Arroyo, and the Palmas del Mar resort. The finest spatial resolution is 8 arc-sec (around 240 meters). A total of 16,000 samples have been obtained using Tsunami-HySEA.

Using the best model, we have obtained mean errors of 1.0 cm and 5.8 sec, and maximum errors of 9.4 cm and 3.3 min. Using ensembles of the three best models for each network, the mean errors have improved to 0.9 cm and 5.0 sec, and the achieved maximum errors have been 8.3 cm and 3.3 min. As a reference, the maximum water height achieved at each point in the samples ranges between 113.3 and 217. 1 cm. The inference times using a Tesla A100 are the same as in the classification problem.

All the presented results for both classification and regression problems have been obtained using the Keras library (Chollet et al, 2015). Both problems have also been successfully implemented and tested using EDDL (C++ version). The early stopping strategy and the reduction of the learning rate on plateau have been implemented in EDDL, since these features are not supported in EDDL by default.

Finally, MPI codes have been developed for both classification and regression problems to run multiple EDDL trainings in parallel, where each training runs in one GPU. At the end of the execution, the best obtained model, which is selected based on the results achieved in the validation set, is saved in ONNX format.

- **ML / AI: Tsunami Forecasting exploiting Regression and Classification Trees**

This activity is aimed at developing machine learning approaches based on regression and classification trees, to model and forecast tsunami simulation results. The goal is to exploit predictive models to (i) reduce the number of simulations and (ii) explore how input values affect output results. A preliminary experimental evaluation has been performed on the dataset of precomputed scenarios for the M6.8 2003 Zemmouri-Boumerdes earthquake and tsunami (one of the project's case studies in the Mediterranean area), retrieved from Selva et al. (2021). The results of this experiment show a good accuracy in maximum wave heights forecasting.

In particular, each simulation data instance related to the considered Tsunami events is composed of 1,119 features: 12 values (simulation input parameters) describing the geometry and the kinematic of faults generating the earthquake (region, magnitude, longitude, latitude, depth of the top, strike, dip, rake, area, length, average slip, probability), and 1,107 values (simulation outputs) corresponding to the maximum heights (hmax) of tsunami waves estimated at several target points in front of the coast close (i.e., 1,107 target points).

The main goal of the machine learning approach we developed has been training a specific predictive model *Mi* for each *i-th* target point (*i=1, …, 1,107*), and exploiting such models to forecast the maximum height of waves brought by the tsunami at the target locations.

The experimental evaluation performed on the Zemmouri-Boumerdes dataset (15,408 records; Selva et al. 2021) provided a good prediction accuracy. The training set and test set have been respectively populated by 10,786 and 4,622 instances (70% and 30% of the whole dataset). The code has been implemented by using the Scikit-learn library (https://scikit-learn.org) and the results (on the test set) have been split by considering differing scenarios w.r.t. on the maximum wave height. The achieved prediction accuracy showed a high accuracy with values ranging from 92% (for wave heights < 0.1 m) to 77% (for wave heights > 2.5 m and < 3.0 m).

Those preliminary results are very promising and further experimental evaluation will be carried out to assess the proposed learning techniques.

Summary of T6.3 for PTF/FTRT WF

The initial and the final implementation of the workflow and its component is reported in table below.

Table 2: Status of the tsunami WF and its components

Workflow	**Status before project**	**Phase I status**
PTF/FTRT	The starting HPC workflow is derived from ChEESE project. It consisted of a non-automized sequence of Matlab, Python, and KUDA scripts, mainly controlled by bash files. No workflow manager was implemented.	PyCOMPSs WF manager integrated in the different steps that compose the tsunami workflow. Full execution in an HPC environment.
Components / Building Blocks	**Initial status**	**Phase I status**

Step 1: Ensemble definition	Matlab scripts derived from Selva et al. (2021), with large ensembles (>10 k events).	Python scripts with enhanced sampling and reduction of the ensemble size. The python calls are embedded in PyCOMPSs.
Step 2: Tsunami simulation	Bash scripts for sequential reading of the scenario ensemble into chunks. Bash scripts for the sequential launch of jobs to the queue system.	PyCOMPSs embedded task to generate chunks and parallel launching of the simulator Tsunami-HySEA. One unique job orchestrates the entire set of simulations. Added Power9 machine at BSC as cluster for end-to-end executions of the workflow.
Step 3: Simulation post-processing	Python scripts extracting parameters from single simulations	PyCOMPSs orchestration by using Ophida framework to manage NetCDF input/outputs files.
Step 4/5: Aggregation and visualization	Matlab scripts derived from Selva et al. (2021)	Python scripts based on OPHIDA output netCDF file
Tsunami Emulator (Block #4)	None	Proof of concept developed
Tsunami Forecasting (Block #4)	None	Experimental evaluation stage

Next Steps of T6.3 for PTF/FTRT WF

At the time of the redaction of this deliverable, the assembly of the steps that contemplate the tsunami workflow is being carried out with the aim of using a single script in PyCOMPSs that orchestrates the end-to-end execution, which is planned for the beginning of M21. (Steps 1 & 2)

The Ophidia team is already working on the workflow, with the next step being testing for one of the use cases. For next iteration, we will work on an efficient cloud storage of model outputs to enable deeper exploitation of simulation results. (Steps 3 & 4)

In common with Steps 1, 2 and 3, one of the next priority steps is to include in the WF the mechanisms for automatic detection of execution errors, which allow it to be relaunched from the moment it occurs, giving it the ability to be resilient to multiple types of failures, a feature that currently does not have.

Regarding the use of ML tools in the workflow, the proofs of concept developed show satisfactory results that will mark the roadmap to follow for their implementation within the workflow that will be incorporated in the next iteration.

4. Workflows codes access

Workflows	
Tsunami PTF/FTRT	https://github.com/carsanlin/workflow-registry/tree/main/iteration1-tsunami-wf
UCIS4EQ	https://gitlab.com/case-geosciences-public/eflows/ucis4eqMockup
Codes	
SeisEnsMan	https://gitlab.com/eflows4hpc/SeisEnsMan
matPTF	https://github.com/INGV/matPTF

5. Conclusions

This deliverable describes the current state of development of the workflows corresponding to Pillar III, based on tasks T6.3 and T6.4, with the improvements in the development and adaptation of workflows, both for earthquakes and tsunamis, by providing a common PyCOMPSs orchestration to easily integrate into HPC environment and turn them into an operational level. It includes the aspects that represents a significant progress in workflow development by comparing the status before and after starting the project.

6. List of Figures and Tables

7. Acronyms and Abbreviations

- D – Deliverable
- BSC – Barcelona Supercomputing Center

- EC – European Commission
- FTRT – Faster Than Real Time
- HPC – High Performance Computing
- KPI – Key Performance Indicator
- MLESmap – Machine-Learning based Estimator for ground motion Shaking maps
- MS – Milestones
- NN – Neural Network
- NEAMTHM18 - NEAM Tsunami Hazard Model 2018
- PSA – Pseudo Spectral Acceleration
- PTF – Probabilistic Tsunami Forecasting
- RF – Random Forest
- SCEC – Southern California Earthquake Center
- UCIS4EQ – Urgent Computing Integrated Services for EarthQuakes
- WF – Workflow
- WP – Work Package

8. References

Álvarez Cid-Fuentes J, Solà S., Álvarez P., Castro-Ginard A., and Badia R. M., “dislib: Large Scale High Performance Machine Learning in Python,” in Proceedings of the 15th International Conference on eScience, pp. 96-105 (2019)

Basili R, Brizuela B, Herrero A, Iqbal S, Lorito S, Maesano FE, Murphy S, Perfetti P, Romano F, Scala A, Selva J , Taroni M, Tiberti MM, Thio HK, Tonini R, Volpe M, Glimsdal S, Harbitz CB, Løvholt F, Baptista MA, Carrilho F, Matias LM, Omira R, Babeyko A, Hoechner A, Gurbuz M, Pekcan O, YalcÄ±ner A, Canals M, Lastras G, Agalos A, Papadopoulos G, Triantafyllou I, Benchekroun S, Jaouadi HA, Ben Abdallah S, Bouallegue A, Hamdi H, Oueslati F, Amato A, Armigliato A, Behrens J, Davies G, Di Bucci D, Dolce M, Geist E, Gonzalez Vida JM, Gonzalez M, Macías Sanchez J, Meletti C, Ozer Sozdinler C, Pagani M, Parsons T, Polet J, Power W, Sørensen W, Zaytsev A (2020), The 28D 6.2 Description of the use cases for Pillar III. Version 1.0 making of the NEAM Tsunami Hazard Model 2018 (NEAMTHM18), Frontiers in Earth Science 9:82, DOI : https://doi.org/10.3389/feart.2020.616594

Basili, R., Brizuela, B., Herrero, A., Iqbal, S., Lorito, S., Maesano, F. E., et al. (2018). NEAM tsunami hazard model 2018 (NEAMTHM18): online data of the probabilistic tsunami hazard model for the NEAM region from the TSUMAPS-NEAM project. Roma: Istituto Nazionale di Geofisica e Vulcanologia (INGV). doi:10.13127/tsunami/neamthm18

Escalante, C., Dumbser, M., Castro M.J. "An efficient hyperbolic relaxation system for dispersive non-hydrostatic water waves and its solution with high order discontinuous Galerkin schemes". Journal of Computational Physics, Volume 394: 385-416, 2019.

Chollet, F. et al. Keras. https://keras.io (2015)

Graves, R., Jordan, T.H., Callaghan, S. et al. CyberShake: A Physics-Based Seismic Hazard Model for Southern California. Pure Appl. Geophys. 168, 367–381 (2011). https://doi.org/10.1007/s00024-010-0161-6

Macías, J., Castro, M.J., Ortega, S., Escalante, C., González-Vida, J.M. "Performance benchmarking of Tsunami-HySEA model for NTHMP's inundation mapping activities". Pure and Applied Geophysics 174 (8), 3147-3183. 2017.

Matias, L. M. et al. Tsunamigenic earthquakes in the Gulf of Cadiz: fault model and recurrence. Nat. Hazards Earth Syst. Sci., 13, 1-13 (2013). https://doi.org/10.5194/nhess-13-1-2013

Monterrubio-Velasco, et al. Source Parameter Sensitivity of Earthquake Simulations assisted by Machine Learning , EGU General Assembly 2021, online, 19–30 Apr 2021, EGU21-5995 (2021) https://doi.org/10.5194/egusphere-egu21-5995.

Intergovernmental Oceanographic Commission, Exercise Caribe Wave/LANTEX 13. A Caribbean Tsunami Warning Exercise, 20 March 2013. Volume 1: Participant Handbook, IOC Technical Series No. 101. Paris, UNESCO (2012).

Selva, J., Lorito, S., Volpe, M. *et al.* Probabilistic tsunami forecasting for early warning. *Nat Commun* **12,** 5677 (2021). https://doi.org/10.1038/s41467-021-25815-w

Tejedor, E., Becerra, Y., Alomar, G., Queralt, A., Badia, R. M., Torres, J., ... & Labarta, J. PyCOMPSs: Parallel computational workflows in Python. The International Journal of High Performance Computing Applications, 31(1), 66-82. (2017)

URLs:

ChEESE Project: https://zenodo.org/record/6376520#.YuDq_HVByo4

CyberShake_Study_15.4: https://strike.scec.org/scecpedia/CyberShake_Study_15.4

CSEM: https://cos.ethz.ch/research.html#par_textimage_1711409697

Deliverable D6.1: https://eflows4hpc.eu/deliverable/d6-1-requirements-on-the-eflows4hpc-software-stack-from-pillar-iii-and-evaluation-metrics-duplicate-1/

Deliverable D6.2: https://eflows4hpc.eu/deliverable/d6-2-development-plan-for-natural-hazards-urgent-computing-workflows-duplicate-1/

dislib: https://dislib.readthedocs.io/en/release-0.7/

EDDL: https://deephealthproject.github.io/eddl/

Ophidia: https://ophidia.cmcc.it/

Tsunami-HySEA: https://github.com/edanya-uma/TsunamiHySEA