



## D6.4 Iteration 2 workflows for urgent computing of natural hazards

Version 1.0

### Documentation Information

<b>Contract Number</b>	9555558
<b>Project Website</b>	<a href="http://www.eFlows4HPC.eu">www.eFlows4HPC.eu</a>
<b>Contractual Deadline</b>	31.08.2021
<b>Dissemination Level</b>	PU
<b>Nature</b>	R
<b>Author</b>	J. de la Puente (BSC)
<b>Contributors</b>	Cedric Bhihe (BSC), Marisol Monterrubio (BSC), Jorge Ejarque (BSC), Jorge Macías (UMA), Carlos Sánchez (UMA), Marc de la Asunción (UMA), Steven J. Gibbons (NGI), Marta Pienkowska (ETH), Eugenio Cesario (DtoK Lab), Salvatore Giampà (DtoK Lab)
<b>Reviewer</b>	Nikolay Koldunov (AWI)
<b>Keywords</b>	urgent computing, earthquakes, tsunamis, natural hazards, WP6, Pillar III



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	Version ready for submission to internal reviewer
V0.3	Reviewer comments addressed
V0.4	Consolidated deliverable
V1.0	Last comments and formatting, ready for submission

## Table of Contents

1. Executive Summary .....	4
2. Introduction.....	4
3. Second iteration of Pillar III workflows .....	4
3.1. The seismic workflow (UCIS4EQ) .....	4
3.1.1. Orchestration with PyCOMPSs .....	9
3.1.2. Deployment with HPCWaaS .....	10
3.2. The Tsunami Workflow (PTF/FTRT) .....	14
3.2.1. Orchestration with PyCOMPSs .....	15
3.2.2. Deployment with HPCWaaS .....	22
3.2.3. Data analytics with Ophidia.....	25
4. Exploratory new components for the workflows .....	27
4.1. MLESMap workflow with dislib and PyCOMPSs .....	28
4.2. Emulators for Tsunami propagation .....	31
4.2.1. Tsunami inundation: Considerations for Training Sets .....	31
4.2.2. Tsunami inundation with EDDL .....	34
4.2.3. Alert Levels .....	34
4.2.4. Maximum Water Height .....	35
4.2.5. Tsunami emulator based on decision-trees .....	35
4.3. PTF based on High resolution HySEA simulations .....	38
5. Workflow access codes .....	40
6. Conclusions.....	41
7. List of figures and tables .....	41
8. Acronyms and abbreviations.....	42
9. References.....	43

## 1. Executive Summary

This deliverable presents Iteration 2 of the earthquake and tsunami workflows belonging to Pillar III: Urgent Computing for Natural Hazards. This iteration is the second iteration both the UCIS4EQ (Earthquake) and PTF/FTRT (Tsunami) workflows, pending the upcoming validation effort (Task 6.5) and integration in workflow repository (Task 6.6). The release fulfills the proposed goal in WP6 description: “... to deliver faster end-to-end runs, with more robust and reliable workflows, and outcomes more usable to potential end-users”. We remark that the final version of the deliverable, fulfilling milestone MS7 of the project, will be the result of the upcoming Task 6.6 due originally in M36 and rescheduled for M38. In order to report on said milestone’s fulfillment, we will add the results of Task 6.6 to Deliverable 6.6. Hence, we will extend its original scope, which was limited to Task 6.7 in the proposal.

## 2. Introduction

The main workflows of Pillar III are UCIS4EQ, which is devoted to performing simulations of earthquakes, and PTF, which simulates tsunamis. They both have undergone a three-phase development process (requirements & design, first development phase, and second development phase) which is now concluded and resulting in the current form of the workflows. For each workflow we describe the additional features added and their impact in the workflow itself. Furthermore, during the development of this activity, new components have been added in an exploratory phase and producing interesting results, using outcomes of the workflows and components of the project’s software stack. We present their current maturity status as well and provide insight in future integration within the main workflows.

## 3. Second iteration of Pillar III workflows

### 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 pieces of the system that communicate with one another and carry out specific tasks (for more details see D6.1). UCIS4EQ has the potential to deliver more accurate short-time reports of the consequences of significant (moderate to large) earthquakes – the workflow rapidly provides synthetic estimates of ground motion parameters, such as peak ground velocity (PGV), peak ground acceleration (PGA), or shaking duration, with very high spatial resolution. Such outcomes can be used to analyze 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).

In this section we describe the work done at Iteration 2, Phase 3 of the project with respect to UCIS4EQ. Our objective is to incorporate developments from WPs 1 to 3 into the workflow in order to improve the resilience and efficiency of the code and enhance the workflow’s monitoring

capabilities. We also enhance the uncertainty treatment by including the previously built high-resolution models to enrich the ensemble for treating uncertainty in urgent computing.

In addition, during Iteration 2 – Phase 3 (M19-M30) four activities have been identified towards increasing the maturity of the UCIS4EQ workflow as marked with orange boxes in Figure 1.

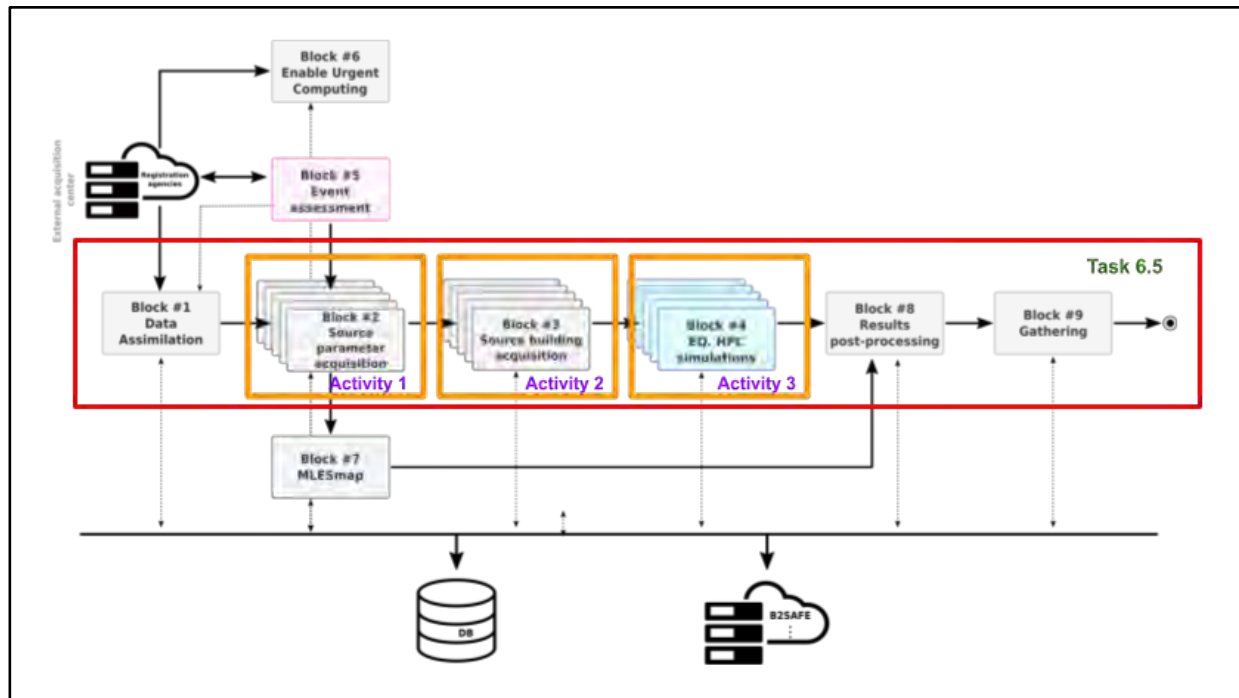


Figure 1. A general overview of the UCIS4EQ workflow. The main 3 activities described in this deliverable are marked with orange rectangles while the validation activity (Task 6.5, M30-36) involves the red rectangle and will be subject of a future deliverable.

### UCIS4EQ Activity 1: SeisEnsMan integration in UCIS4EQ as a prototype

A new module has been developed to implement a new method to manage the uncertainty on the seismic source parameters through the definition of an ensemble of simulations, aiming at a coherent management for both seismic and tsunami urgent computing. SeisEnsMan, the module aiming at managing the ensemble in UCIS4EQ, introduces several specializations required to specify the seismic sources, in our case sliding geological faults. New python tools were developed to test the new parametrizations and monitor the convergence of the uncertainty quantification, adopting standard Ground Motion Prediction Equations (GMPEs) models to evaluate the potential seismic impact of the different sources.

SeisEnsMan was embedded into a Docker container to fulfill the requirements of integration within the UCIS4EQ microservices design structure. The integration of this new ensemble method implied the modification of UCIS4EQ codes including different components and also a modification on the so-called *WorkflowManager* service.

For a complete SeisEnsMan integration in UCIS4EQ, the Docker image must be incorporated as a new Service orchestrated by PyCOMPSs. As is shown by the red dotted lines in Figure 3 the listener alert assimilated in Building Block 1 must call the SeisEnsMan service and pass the outputs to Building Block 2 where the source parameters are set. This action could complement and not

substitute the current method used in UCIS4EQ, namely statistical CMT (Monterubio et al., 2021, [S9]). The basic differences between these methods (statistical CMT and SeisEnsMan) is illustrated in Figure 2. On one hand, the statistical CMT method populates the ensemble with different CMT's preserving the hypocentral coordinates (latitude, longitude and depth), and the magnitude of the earthquake. On the other hand, SeisEnsMan populates the ensemble with variations of the hypocentral location, magnitude, CMT, fault length and fault width, sampling them from the uncertainty on those parameters provided by the seismic monitoring systems and the hazard knowledge of the source area (see Selva et al. 2021 [S1]). Another difference is the size of the ensemble. While the statistical CMT depends upon historical catalogs to populate the ensemble, SeisEnsMan generates ensembles of hundreds to thousands of sources to cover the potential natural variability of seismic events (Stallone et al. 2023, [S13]). This imposes pros and cons for either method, which can thus be seen as complementary methodologies. For example, for regions where the historical seismicity is scarce the SeisEnsMan module can help increase the number of possible sources. The current version of SeisEnsMan in the repository is limited to the Mediterranean region, which constrains the possible case studies at present time. A worldwide version can be derived from Taroni and Selva (2021) [S4] and Folch et al. (2023) [S2], but we have decided upon focusing on the Mediterranean, where more accurate prior distributions have been derived (Selva et al. 2016 [S5], 2021[S1]; Basili et al. 2021 [S6]).

Along the Iteration 2 - Phase 3 of the project, a first prototype of the SeisEnsMan module integrated in UCIS4EQ has been set up and successfully executed. In this exercise the outputs generated by SeisEnsMan for the Samos Izmir 2020 earthquake were manually uploaded in the B2DROP repository, and read at runtime by UCIS4EQ (green line of Figure 3). UCIS4EQ, using the new ensemble, was successfully executed in MareNostrum4.

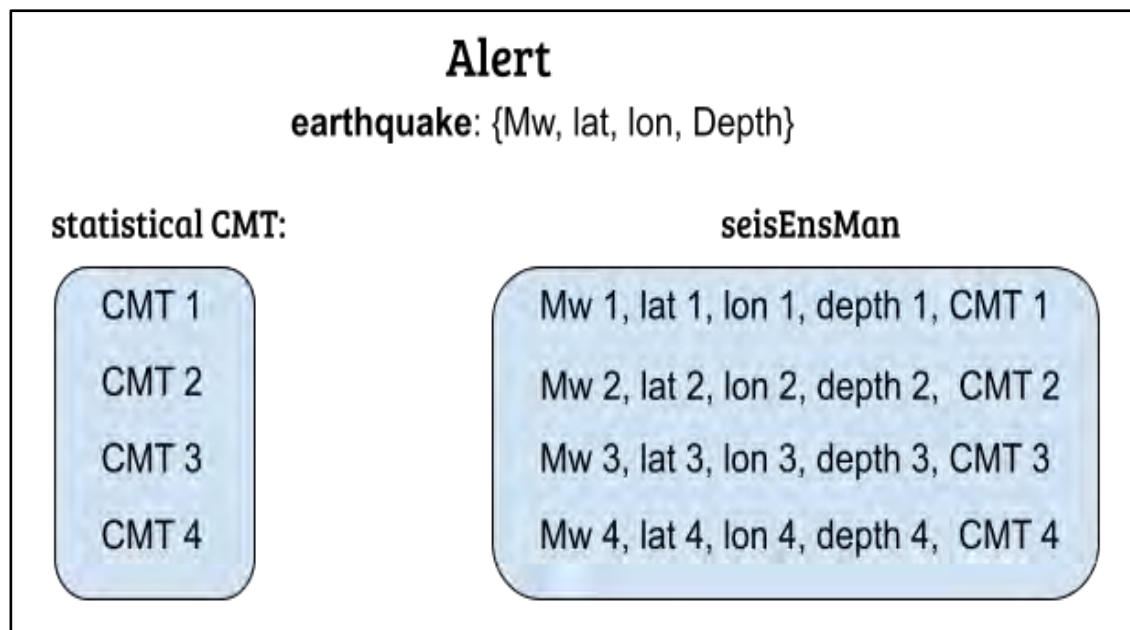


Figure 2. Schematic differences between the ensemble methods used in UCIS4EQ, the statistical CMT (Monterubio et al., 2021) and SeisEnsMan. The details are described in the text.

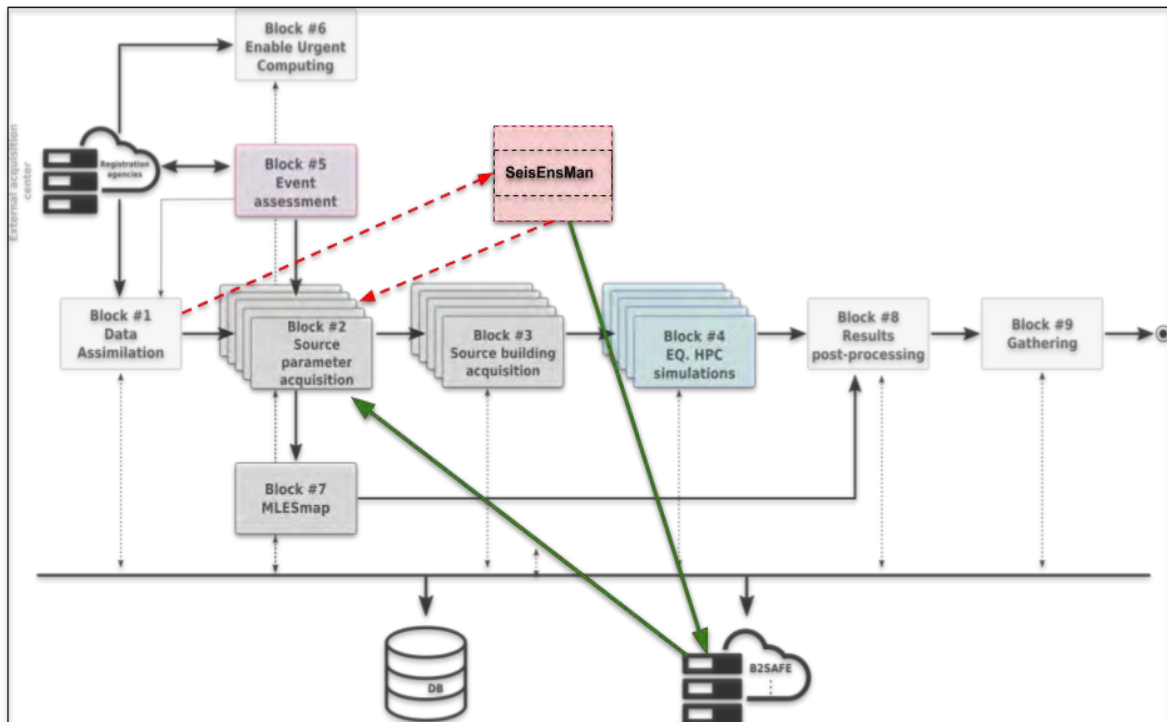


Figure 3. Representation of the SeisEnsMan module and its interaction with the UCIS4EQ building blocks. The red dotted lines and the green lines represent the two strategies to integrate this ensemble method in the UCIS4EQ workflow.

### UCIS4EQ Activity 2: Producing new Earth models for the regions of interest defined in Task 6.2

This activity is fully reported in the Deliverable 6.5, with the same due date as this deliverable.

### UCIS4EQ Activity 3: Salvus wrapper update

One of the main objectives is to continue improving the functions that are used to provide Salvus<sup>1</sup> with suitable parameters to run, i.e. `salvus_wrapper`, to pre-process the incoming data from Building Blocks 2 and 3. In particular, it is crucial to update it with the new functionalities provided in the latest Salvus releases in order to optimize the model generation and the preparation of the input data to launch Salvus simulations. Currently, some updates have been tested offline, outside of the UCIS4EQ, and work is ongoing to merge those changes with the `salvus_wrapper` located at the HPC. For example, currently Salvus flow incorporates the functions to transform the elliptic to geocentric latitude coordinates used to modify if needed the source's and receivers' locations in the correct format for the mesh. Prior to that a Python script was used to carry out that transformation. Integration methods (or mechanisms) for the topography, bathymetry, and velocity models were updated in the Salvus 0.12.13 version, in order to make it more user-friendly.

### UCIS4EQ Activity 4: Towards the validation of UCIS4EQ

To start setting the drill runs of representative cases in production-like environments (Task 6.5, M30-M36) we have started to prepare locally all the input data required by UCIS4EQ for the Mexican earthquake, as proposed in D6.2. This work was carried out in close collaboration with the Seismological Mexican Service SSN (<http://www.ssn.unam.mx/>), to ensure a reasonable

<sup>1</sup> <https://mondaic.com/product/>



maturity in future drill runs, towards establishing UCIS4EQ as an operational HPC seismological service.

The most important inputs for UCIS4EQ in a new region are: the set of receivers (Figure 4 a), updated historical CMT catalog (Figure 4 b), magnitude-area relations (Figure 4 c) and computational mesh for different frequencies (Figure 4 d). Final efforts along this activity will be conducted as part of Task 6.5, due in M36.

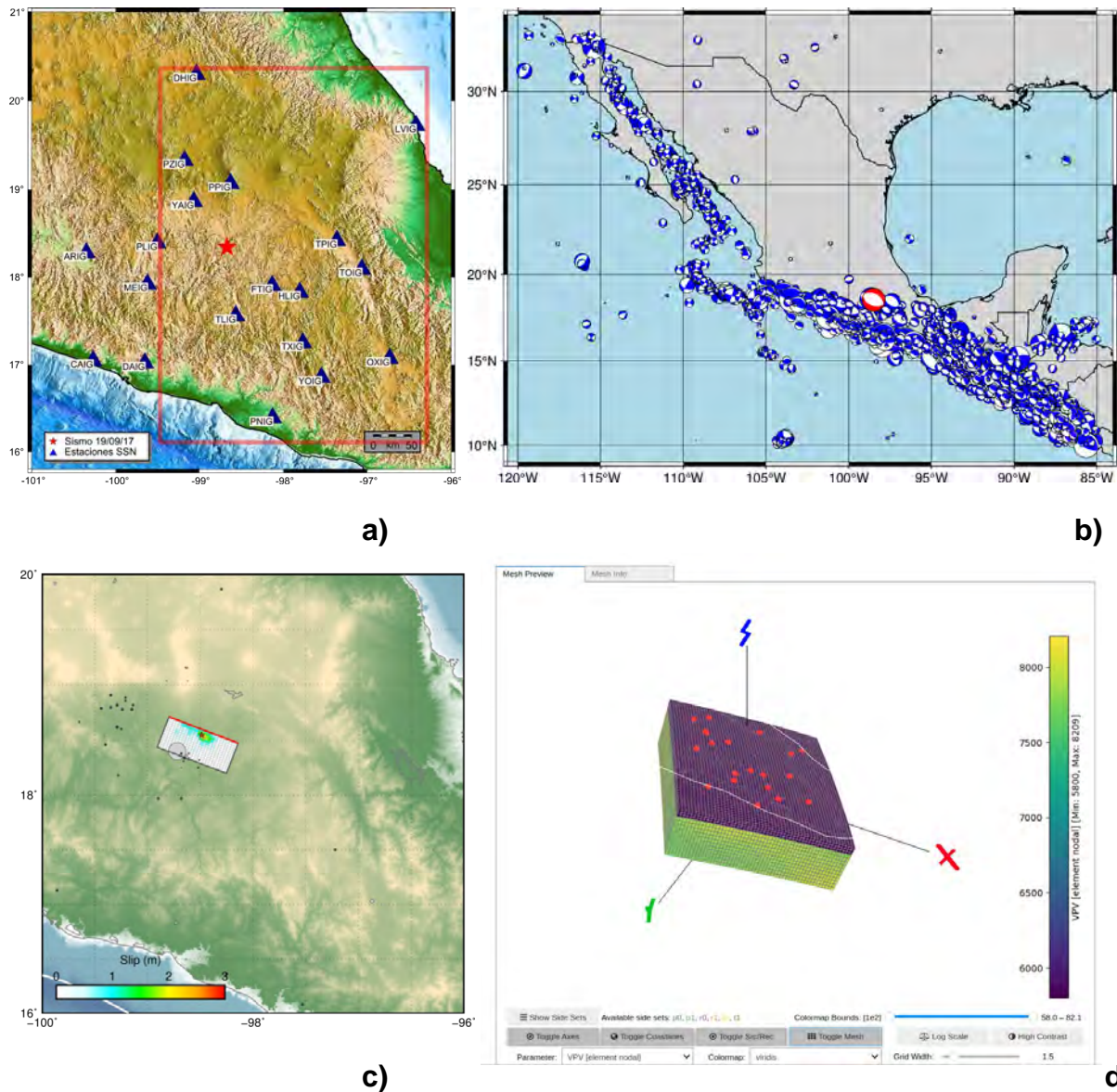


Figure 4. Required information to set the new Mexican earthquake towards Task 6.5. a) Set of receivers b) Historical CMT catalog, c) magnitude-area relations and rupture generator parameters that can lead to a finite fault model, and d) the computational mesh



### 3.1.1. Orchestration with PyCOMPSs

The first iteration of the project has demonstrated the integration of PyCOMPSs as a workflow manager to orchestrate the microservices in UCIS4EQ. In particular, the microservices corresponding to the building blocks are shown in the red rectangle of Figure 1, which include executing simulations for several versions of the fault rupture, each stemming from a different CMT estimate and/or seed for the rupture generator, i.e. several runs. The integration of PyCOMPSs in UCIS4EQ gives the capacity to the workflow to fulfill the proposed execution requirements described in Table 1.

To orchestrate the microservices calls, PyCOMPSs has been extended with the `@http` decorator. This is an example of collaboration between WP6, who required the feature, and WP2 that implemented it during the iterations of the project. It allows developers to define a task that performs an HTTP request as indicated in Figure 5.

```
@on_failure(management='CANCEL_SUCCESSORS')
@http(request="POST", resource="Graves-Pitarka", service_name="slipgen",
      payload='{ "event" : {{alert}}, "id" : {{event_id}}, "CMT" : {{cmt}}, \
                "trial" : "{{path}}", "region": {{region}}, "setup" : {{setup}}, \
                "resources" : {{resources}}, "ensemble" : "{{ensemble}}" }',
      produces='{"result" : "{{return_0}}"}')
@task(returns=1)
def compute_graves_pitarka(event_id, alert, path, cmt, region, setup, resources, ensemble):
    pass
```

Figure 5. Task definition with PyCOMPSs for an asynchronous microservice invocation.

As in a normal task the developer has to specify the task parameters. In addition it can also indicate where the task parameters will be included or gathered from the HTTP request. It can also be combined with the failure management mechanism to retry, ignore or cancel the successor tasks to manage the failure that can appear in the workflow. This feature also reduces the amount of code written by the developer: In particular it obviates the need to emit HTTP requests with the corresponding Python libraries. Finally orchestration with PyCOMPSs also replaces our original code's asynchronous execution of the Salvus simulations. In this case, the asynchronous execution management code is not required because the defined tasks are asynchronously executed.

During this second iteration, different end-to-end executions of UCIS4EQ were executed on CSCS' Piz Daint (GPUs) and Marenstrum4 (CPUs) for high-frequency simulations up to 5Hz. For this purpose, we exploited two use cases proposed in D6.2:

- Mediterranean Sea: the 2017 M6.6 Kos-Bodrum earthquake, 120 km x 100 km
- Iceland: the 2000 (June 21) M6.5 SISZ earthquake, 135 km x 85 km domain

The results of these successful executions have been presented at international conferences during this third phase (Monterrubio et al., 2022 [S7]; Monterrubio et al., 2023 [S8])

Apart from the PYCOMPSs orchestration of microservices, we also created PyCOMPSs workflows to orchestrate different HPC executions in this second iteration. In the original UCIS4EQ, every execution in the HPC system was performed in a separate service call and each call was submitting

a job to the HPC system, with its corresponding overhead. Moreover, as every system has its own job scheduler, the original UCIS4EQ workflow implements a set of adaptors to submit the job in the HPC schedulers of every machine. As PyCOMPSs is supporting several HPC schedulers and has the same execution interface, these adaptors are not required anymore.

```
@container(engine="SINGULARITY", image="$SLIPGEN_IMAGE", options="-e --bind {{workingdir}}:/workspace/ --pwd /workspace")
@binary(binary="/opt/scripts/launcher.sh", args="-o rupture --dt {{dt}} -v {{fk_file}} -s {{input_src}}", working_dir="{{workingdir}}")
@task(input_src=FILE_IN, fk_file=FILE_IN, workingdir=DIRECTORY_INOUT)
def slipgen(input_src, dt, fk_file, workingdir):
    pass

@task(input_data=FILE_IN, rupture=DIRECTORY_IN, salvus_setup=FILE_IN, working_dir=DIRECTORY_INOUT)
def salvus_prepare(input_data, rupture, salvus_setup, working_dir):
    rupture_file = rupture + "/scratch/outdata/rupture/rupture.srf"
    os.chdir(working_dir)
    pre_process(input_data, rupture_file, salvus_setup, working_dir)

@mpi(runner="mpirun", binary="$SALVUS_BINARY", args="compute {{prepare_path}}/sylvus_input_rupture.toml", processes="$SALVUS_PROCESSES",
processes_per_node= "$SALVUS_PPN", working_dir="{{working_dir}}")
@task(prepare_path=DIRECTORY_IN, working_dir= DIRECTORY_INOUT)
def salvus_run(prepare_path, working_dir):
    pass

@task(UC_input = FILE_IN, salvus_setup= FILE_IN, grid_coordinates= DIRECTORY_IN, simu_folder=DIRECTORY_IN, output_path=DIRECTORY_INOUT)
def salvus_post(UC_input, salvus_setup, grid_coordinates, simu_folder, output_path):
    process_outputs_grid( UC_input, salvus_setup, grid_coordinates, simu_folder, output_path=output_path)

if __name__ == "__main__":

    slip_id, input_path, salvus_setup, slip_input_src, region_fkId, dt = parse_arguments()
    slipgen_dir, salvus_wrapper_dir, salvus_dir, salvus_post_dir = create_output_dirs()

    slipgen(slip_input_src, dt, region_fkId, slipgen_dir)
    salvus_prepare(input_path, slipgen_dir, salvus_setup, salvus_wrapper_dir)
    salvus_run(salvus_wrapper_dir, salvus_dir)
    salvus_post(input_path, salvus_setup, salvus_wrapper_dir, salvus_dir, salvus_post_dir)
```

Figure 6. Code Snippet of the HPC workflow with PyCOMPSs

Figure 6 shows a code snippet with the HPC Workflow implemented in PyCOMPSs. This workflow describes 4 tasks: the *slipgen* which runs the slip generation using a singularity image: the *sylvus\_prepare* and *sylvus\_post* which executes the Sylvus preprocessing and postprocessing as normal python tasks, and the *sylvus\_run* which performs the simulation with Sylvus defined as an MPI application. The logic of the workflow is implemented as a simple python script where the different tasks are called as normal python methods. This workflow is called from the microservices workflow which submits the HPC Workflow using the PyCOMPSs queuing scripts which already supports different schedulers. This is executed for each run required by the UCIS4EQ microservices. Similarly, we have created another workflow which aggregates the results from all the runs, executes the swarm postprocessing and generates the final plots.

As part of Task 6.5 the new PyCOMPSs implementation will be tested in a drill run for the 2017 M7.1 Puebla earthquake proposed in D6.2.

### 3.1.2. Deployment with HPCWaaS

One of the main outcomes of the project is the HPC Workflow as a Service (HPCWaaS) platform to facilitate the reusability of these complex workflows in federated HPC infrastructure. We started the integration of the HPC part of the UCIS4EQ workflow with the HPCWaaS platform, in view to

finish it during the last phase of the project. To better appreciate the scope of this integration, we first propose to describe TOSCA using the Alien4Cloud software, as in Figure 7.

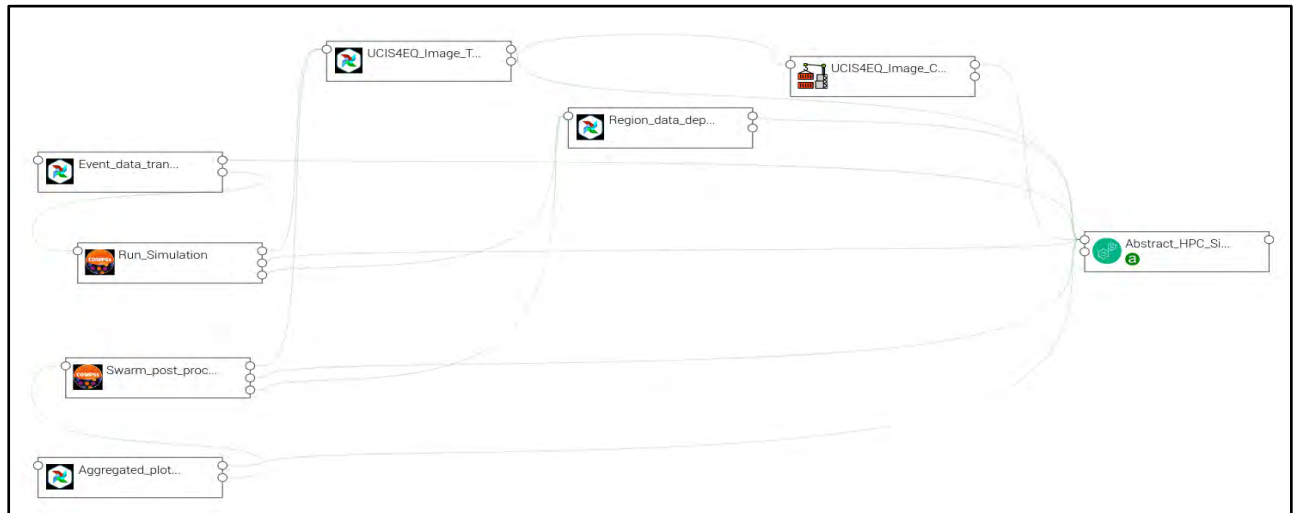


Figure 7. Graph of TOSCA pipelines used by UCIS4EQ

This description shows the different TOSCA components involved in the deployment and execution of the workflow.

- Setup phase:  
The *Abstract\_HPC\_Site* component defines the properties (login node address, CPU architecture, supported container engine, ...) of the HPC system where we mean to deploy and run the workflow. Every component that needs to be aware of the target HPC system depends on *Abstract\_HPC\_Site*.
- Deployment phase:  
The *UCIS4EQ\_Image\_Creation* component implements the interaction with the eFlows4HPC Container Image Creation (CIC) service to build a container Image including all the software components required for the workflow.  
The *UCIS4EQ\_Image\_Transfer* component implements the interaction with the Data Logistics Service (DLS). It depends on the *UCIS4EQ\_Image\_Creation* component because it has to know the URL of the generated container image in order to perform the image deployment.  
The *Region\_Data\_deployment* component interacts with the DLS, but in this case it is configured to download the data of a region (maps, etc.) from the data-set stored in the UCIS4EQ B2DROP repository.

At execution, we defined two TOSCA components and two data pipelines.

- The TOSCA components are *Run\_Simulation* and *Swarm\_post\_processing*. They submit the execution of the HPC workflow described in the previous section.
- The data pipelines are for the stage-in of the event data to simulate (*Event\_data\_transfers*) and for the stage-out to upload the generated plots at the end of the swarm post-processing workflow (*Aggregated\_plots\_upload*).

The status of this integration at the end of the second iteration is the following:

- The different data pipelines to download data from/to the UCIS4EQ B2DROP were implemented and tested to download them to the MN supercomputing facility.
- We manually created a container image with Salvus and other required dependencies for MN-only.
- The different HPC workflows were implemented and their execution tested with the MN-only container image

On-going work includes the integration of the Salvus installation and of the Container Image creation. Salvus is a proprietary/commercial software edited by Mondaic. A license must be secured by a user for the software download to become possible. In practice the software editor, Mondaic, sets up a licensed user account and, in so doing, gives that authorized user the credentials required for download. The software download center offers authorized users software in binary form. We are in the process of integrating the Salvus installation into the Container Image Creation scripts. This involves requesting the download of the Salvus binary package adapted to the target platform as well as automating the full image creation and deployment processes.

The last step of the integration will be the automation of the workflow deployment with the Alien4Cloud interface and its execution invoking the HPCWaaS Execution API from the UCIS4EQ microservices.

To conclude the work carried out for UCIS4EQ in this phase of the project it is also important to assess the status of the requirements proposed in D6.1. In Table 1 we include the status of the requirements in Phase 3 for the UCIS4EQ. A detailed description of each requirement can be found in Table 5 in the D6.1 document. The Priority column indicates the level of necessity of the given requirement for the Pillar III workflows. By the final release (M38) most requirements will be met.

Table 1. Status of the execution requirements in the UCIS4EQ Workflow of Pillar III (D6.1) and the current status at the Iteration 2 - Phase 3 (M19-M30).

ID	Name	Priority (must, should, may)	Status at Phase 3	Comment on the current status
1	Urgent computing access	must	Ongoing work in Task 6.7 (M6-30)	A first attempt to provide a protocol to enable the Urgent computing access mode in Tier-0 and Tier-1 machines will be delivered in D6.6 at the end of this project.
2	Data accessibility	should	Ongoing work	The data accessibility is provided by the Data Logistic Service through a pipeline to move the data required in the HPC and the results. In the final iteration of the workflow we employ the B2DROP repository. This requirement is also important for a correct deployment of the workflow in other HPC centers.
3	Data replication	must	Ongoing work	The redundancy of data required by UCIS4EQ in the different phases must be also guaranteed by the DLS pipelines in the deployment status. Currently the data for the different test cases are replicated at two distinct HPC centers, MareNostrum 4 and Piz

				Daint. Some data is also replicated as well on the B2DROP service (see also ID9 in this table).
4	Execution Robustness	must	Implemented	This requirement is fulfilled by means of the PyCOMPSs workflow manager which provides support to fault tolerance during the workflow execution including checkpoints or retries.
5	Infrastructure interoperability	must	Implemented	The infrastructure interoperability has been successfully implemented by means of the deployment system and also the introduction of PyCOMPSs to orchestrate the computations performed at different infrastructures (HPC clusters and external servers).
6	Portability and Reusability	may	Ongoing work	The workflow can now be safely deployed by means of TOSCA and thus can be ported to a variety of HPC systems. The components have been containerized in order to allow for their substitution or reuse
7	Streaming Data Source	must	Ongoing work	The acquisition of real-time streaming data sources should be developed in Building Block 5. A first prototype is currently being developed that includes a new service orchestrated by PyCOMPS to meet this requirement.
8	Integrated workflow manager	must	Implemented	PyCOMPSs is fully integrated within UCIS4EQ as is described in section 3.1.1 of this document.
9	Integration with permanent storage	must	Partially Implemented	Currently the storage of the data required by the workflow is split between repositories at HPC centers for large files (MN4 and Piz Daint), as well as B2DROP for lighter configuration files. The final results of the workflow are automatically uploaded to B2DROP, while the intermittent results (such as full simulation outputs) remain on the HPC cluster. Choice and integration of storage in long-term repositories is currently analyzed. (see also ID3 in this table)
10	Inference with online/offline ML models	must	Partially implemented	MLESmap provides ML models that can be used in UCIS4EQ within Building Block 8. The ML models are prepared “offline” (that is, not in an urgent manner) and must be ready for an online application with rapid queries. The first test is currently being implemented in Iceland, but remains a non-integrated feature of UCIS4EQ, i.e. standalone, within the framework of the project.
11	DA integration	may	Not implemented	The UCIS4EQ post-processing stage requires the management of large HDF5 output files obtained in each simulation of the ensemble of sources in order to gather the results in a final single file. The management of such HDF5 files is not provided by any data analytics tool in the project. Hence, our solution uses python scripts to carry out this task.
12	Workflow	should	implemented	Malleability is a characteristic inherently provided



	malleability			by the PyCOMPSs implementation of the HPC components of UCIS4EQ. It has not been tested, as applications so far did not need malleable resource management, but should not pose a problem.
--	--------------	--	--	--

### 3.2. The Tsunami Workflow (PTF/FTRT)

The Tsunami Workflow (PTF – *Probabilistic Tsunami Forecasting* / FTRT - *Faster Than Real Time*) seeks to provide a forecast of tsunami impact following a large offshore or near-shore earthquake soon after its occurrence. The uncertainty in the source is dealt with by considering a (potentially large) ensemble of earthquake scenarios, combining them to provide a Probabilistic Tsunami Forecasting. For each such scenario, an efficient numerical simulation needs to be performed in which the impact on the coastlines of interest is calculated. These *Faster Than Real-Time* numerical simulations are conducted using Tsunami-HySEA, a dedicated numerical propagation and inundation model specifically designed for tsunamis, developed by the EDANYA group of the University of Málaga. The model employs cutting-edge finite volume methods, which offer a powerful combination of robustness, reliability, and precision within a single GPU-based framework, enabling simulations to run faster than real-time. Tsunami-HySEA has undergone extensive testing and validation [hysea1-hysea5], ensuring its accuracy and suitability for tsunamis. Additionally, it has been rigorously validated and verified in accordance with the standards set by the National Tsunami Hazard and Mitigation Program (NTHMP) of the United States [hysea6-hysea8]. This comprehensive validation process ensures the model's capability to provide trustworthy and accurate results, making it a valuable tool for the study of tsunamis and of their potential impacts on coastal regions.

The aim of this task is (a) to adapt and further develop the workflow for Probabilistic Tsunami Forecasting (PTF, Selva et al. 2021, S1; Folch et al. 2023, S2) to include dynamic and iterative features using the eFlows4HPC software stack, and (b) to improve the computational and the scientific performance of existing PTF.

The original non-automated workflow consisted of the three main sequential stages: (i) seismic source ensemble initialization, developed in Matlab, (ii) multiple Faster Than Real-Time (FTRT) numerical tsunami simulations using the multi-GPU software T-HySEA, and (iii) post-processing with hazard aggregation, again developed in Matlab, enabling Probabilistic Tsunami Forecasting (PTF). The new dynamic and iterative approach to the PTF workflow has been fully integrated with eFlows4HPC software stack, with all components of pre- and post-processing in Python, and it includes significant added functionalities as (1) a near-real-time source-estimation manager to initialize PTF ensemble forecast based on earthquake data using a new sampling procedure to reduce the required number of simulations, (2) a more efficient cloud storage of model outputs to enable deeper exploitation of simulation results, and (3) increased event diagnostics to monitor potential simulation failures, (4) a cyclical simulation scheme to enable the production of early results based on the first available simulations and to enable future integration of AI tools as surrogate tsunami simulation models, and (5) a new module to manage the potential update of the PTF's ensemble based on data that can be made available during the workflow execution (e.g. moment tensor solutions, tsunami wave gauge data etc.). Subtasks 2 to 5 were conducted Iteration 2, Phase 3.



### 3.2.1. Orchestration with PyCOMPSs

With the aim of making it flexible in its future developments and adaptable to different systems requirements, the workflow initially written in Matlab prior to the eFlows4HPC project, was progressively converted to Python and entirely modularized. This modularization greatly facilitated its inclusion into a PyCOMPSs framework.

A PyCOMPSs application is composed of tasks, which are methods annotated with decorators following the PyCOMPSs syntax. The tasks can be parts of the python script, but also of other types of binaries and external codes. They are called successively but also in parallel in a main python script that manages the full workflow.

The structure of the workflow is as presented in Figure 8:

(1) The Step1 PTF ensemble manager is called as the first task and generates a list of scenarios (Okada fault model parameters) with associated probabilities representing the likelihood that any given scenario will resemble the ongoing event (Selva et al. 2021, [S1]). From this list, a large ensemble of scenarios is generated by sampling from this probability distribution (Cordrie et al 2023a,b [ens1,ens2]). Several bash script files prepare the Tsunami-HySEA configuration files and simulation folder structure based on the Step1 output list of parameters.

(2) The Tsunami-HySEA code is called in a loop with a double parallelization. PyCOMPSs optimizes the repartition of the jobs (each loop representing a job) between the active nodes and HySea works with an intrinsic MPI-MC parallelization that allows the parallel calculation of 4 simulations (4 threads for one GPU node), see Figures 9 and 10. Inside the loop, and sequentially to the end of each simulation, a first post-processing task is performed and communicates to the rest of the on-going processes the completion of the post-processing by updating a commutative file.

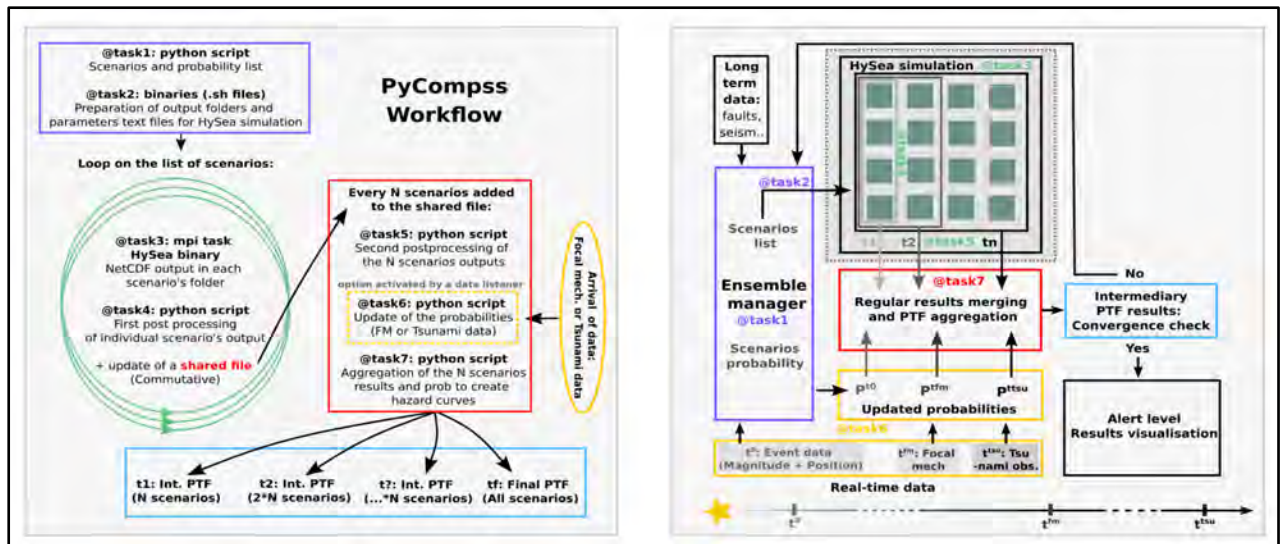


Figure 8. Orchestration of the PTF workflow using PyCOMPSs: Left: definition and order of the tasks. Right: schematic diagram of the workflow with the time-line.

## PyCOMPSS tasks and functions definition

```
# Required paths
tsunamiHySEA_bin= HySEA_dir + "/bin/TsunamiHySEA"
load_balancing_bin= HySEA_dir + "/bin_lb/get_load_balancing"

simulBS_bin = install_dir+"/Code/scripts/Step2_config_simul.sh"
config_bin = install_dir+"/Code/scripts/Create_config.sh"

# GPU size: 1 gpus per node x gpus_per_exec
gpus_per_node=os.environ.get("PTF_GPUS_NODE",4)
gpus_per_exec=os.environ.get("PTF_GPUS_EXEC",4)

##### PyCOMPSS TASKS #####

## TASK FOR STEP1 ##
@task(config_file=FILE_IN, returns=1)
def step1_func(args, config_file, seistype, sim_files_step1):
    args.cfg=config_file
    run_step1_init(args,sim_files_step1)
    return sim_files_step1 + "/Step1_scenario_list_"+seistype+".txt"

@binary(binary=config_bin)
@task(config_file=FILE_OUT)
def build_config(config_template, config_file, data_dir, files_step2, par_file, kag, tsu, event_id):
    pass

## TASKS FOR STEP2 ##
## execution of simulator HySEA using pycompss: (test)
@constraint(processors=[{'processorType':'CPU', 'computingUnits':1},
                        {'processorType':'GPU', 'computingUnits':1}])
@mpi(binary=tsunamiHySEA_bin, args="{file_in}", runner="mpirun", processes=gpus_per_exec, processes_per_node=gpus_per_node,working_dir="{wdir}")
@task(file_in=FILE_IN,returns=1)
def mpi_func(file_in, wdir):
    pass

@binary(binary=simulBS_bin, working_dir="{wdir}")
@task(sim_files_step2=FILE_OUT)
def build_structure(seistype, grid, hours, group, sim_files_step2, load_balancing, pois_ts_file, sim_template_file, sim_events_files, wdir):
    pass

#@task(log_file=FILE_OUT)
@task(returns=1)
def extract_ts(out_ts,depth_file,ptf_file,simwdir,centinel):
    step2_extract_ts(out_ts,depth_file,ptf_file,simwdir)

#@task(log_file=FILE_OUT)
@task(returns=1)
def create_ptf_input(ptf_files,out_path,depth_file,log_file):
    step2_create_ptf_input(ptf_files,out_path,depth_file,log_file)

@task(ptf_files=COMMUTATIVE, config_file=FILE_IN)
def append_and_evaluate(ptf_files, ptf_file, args, config_file, sim_files_step1, out_step2_path, out_update_path, out_final, depth_file, log_file, sim_pois_ts, num_sims, kag, tsu, result_ext):
    args.cfg = config_file
    ptf_files.append(ptf_file)
    if (num_sims != 0) and (len(ptf_files) % num_sims == 0):
        step2_create_ptf_input(ptf_files, out_step2_path, depth_file, log_file)
        if kag>0:
            run_step_kagan(args,sim_files_step1,out_update_path)
            sim_files_input=out_update_path
        elif tsu>0:
            run_step_mare(args, sim_files_step1, out_update_path, sim_pois_ts, ptf_files)
            sim_files_input=out_update_path
        else:
            sim_files_input=sim_files_step1
            run_step3_init(args, sim_files_input, out_final, sim_pois_ts, ptf_files)

def build_steps_dirs(exec_dir, seistype):
    files_step1 = exec_dir + "/ptf_local_step1/"
    os.makedirs(files_step1)
    files_step2 = exec_dir + "/ptf_local_step2/"
    os.makedirs(files_step2)
    intermediate_files = exec_dir + "/ptf_local_update/"
    os.makedirs(intermediate_files)
    files_step3 = exec_dir + "/ptf_local_step3/"
    os.makedirs(files_step3)
    step2_folder = exec_dir + "/Step2_"+seistype
    os.makedirs(step2_folder)
    return files_step1, files_step2, files_step3, intermediate_files
```

Figure 9. PTF workflow PyCOMPSSs script: PyCOMPSSs tasks and functions' definition.



## PyCOMPSS workflow main script

```
##### Main SCRIPT #####

if __name__ == '__main__':

    # reading arguments

    args = parse_ptf_stdin()
    exec_dir = args.run_path
    data_dir = args.data_path
    template_dir = args.templates_path
    par_file = args.parameters_file
    # why are there 2 args for the par_file in run_bsc_mod ?
    if (exec_dir is None) or (data_dir is None) or (template_dir is None) or (par_file is None):
        print("One of these parameters is missing: run_path, data_path, templates_path, par_file")
    else:
        hours = args.hours
        # the arg hours is missing in the run_bsc_mod.sh
        group_sims = int(args.group_sims)
        # same
        print("Kagan: " + args.kagan_weight)
        print("Mare: " + args.mare_weight)
        kag = int(args.kagan_weight)
        tsu = int(args.mare_weight)
        #cfg_file = args.cfg
        seistype = args.seistype
        event_id = args.event
        args.event = data_dir + "/earlyEst/" + event_id + "_stat.json"

        files_step1, files_step2, files_step3, intermediate_files = build_steps_dirs(exec_dir, seistype)

        ### Building Configuration
        config_file = exec_dir + "/ptf_main.config"
        config_template = template_dir + "/Step1_config_template_mod.txt"
        build_config(config_template, config_file, data_dir, files_step2, par_file, kag, tsu, event_id)

        ### creation of the scenario ensemble ###
        sim_step1_events_file = step1_func(args, config_file, seistype, files_step1)

        depth_file = data_dir + "/regional_domain/bathy_grids/regional_domain_POIs_depth.dat"
        grid_file = data_dir + "/regional_domain/bathy_grids/regional_domain.grd"
        pois_file = data_dir + "/regional_domain/POIs.txt"
        sim_template_file = template_dir + "/Step2_parfile_tmpl.txt"
        log_file = files_step2 + "/Step2_" + seistype + "_failed.txt"
        sim_pois_ts = exec_dir + "/Step2_ts.dat"
        sim_files_step2 = exec_dir + "/sim_files.txt"

        ### Preparation of the files for HySEA simulation ###
        build_structure(seistype, grid_file, hours, gpus_per_node, sim_files_step2, load_balancing_bin, p
ois_file, sim_template_file, sim_step1_events_file, exec_dir)
        compss_wait_on_file(sim_files_step2)

        ptf_files = []
        sims = 0
        ### HySEA simulations ###
        with open(sim_files_step2) as f:
            for line in f:
                # load balancing
                line = exec_dir + "/" + line.strip()
                print("Submitting execution for " + line)
                result = mpi_func(line, exec_dir)

                with open(line) as fsim:
                    print("Entering fsim :", line)
                    for simline in fsim:
                        sims = sims + 1
                        simline = simline.strip()
                        print("Entering simline :", simline)
                        simwdir = os.path.dirname(simline)
                        print("sim dir", simwdir)
                        out_ts = exec_dir + "/" + simwdir + "/out_ts.nc"
                        ptf_file = exec_dir + "/" + simwdir + "/out_ts_ptf.nc"
                        ts_result = extract_ts(out_ts, depth_file, ptf_file, simwdir, result)
                        append_and_evaluate(ptf_files, ptf_file, args, config_file, files_step1, files_st
ep2, intermediate_files, files_step3, depth_file, log_file, sim_pois_ts, group_sims, kag, tsu, ts_result)
                    fsim.close()
                f.close()

        # compute a final evaluation if final group was not multiple of group_sims
        if (group_sims == 0) or (sims % group_sims != 0):
            append_and_evaluate(ptf_files, ptf_file, args, config_file, files_step1, files_step2, intern
ediate_files, files_step3, depth_file, log_file, sim_pois_ts, 1, kag, tsu, ts_result)
```

Figure 10: PTF workflow PyCOMPSS script: workflow main script

(3) A listener to this commutative file authorizes further calculations every time a specific (predefined) number of simulations is reached. These calculations include early stage post-processing, that is a post-processing task performed with the available simulation results, and convergence monitoring.

(4) Optionally, at this stage of the workflow, an update of the ensemble based on new incoming data can be performed.

(5) The post-processing calculates the PTF hazard curves and stores them in specific folders.

At execution time, the runtime builds a task graph (see Figure 11) that takes into account the data dependencies between tasks, and from this graph schedules and executes the tasks in the distributed infrastructure, taking also care of the required data transfers between nodes.

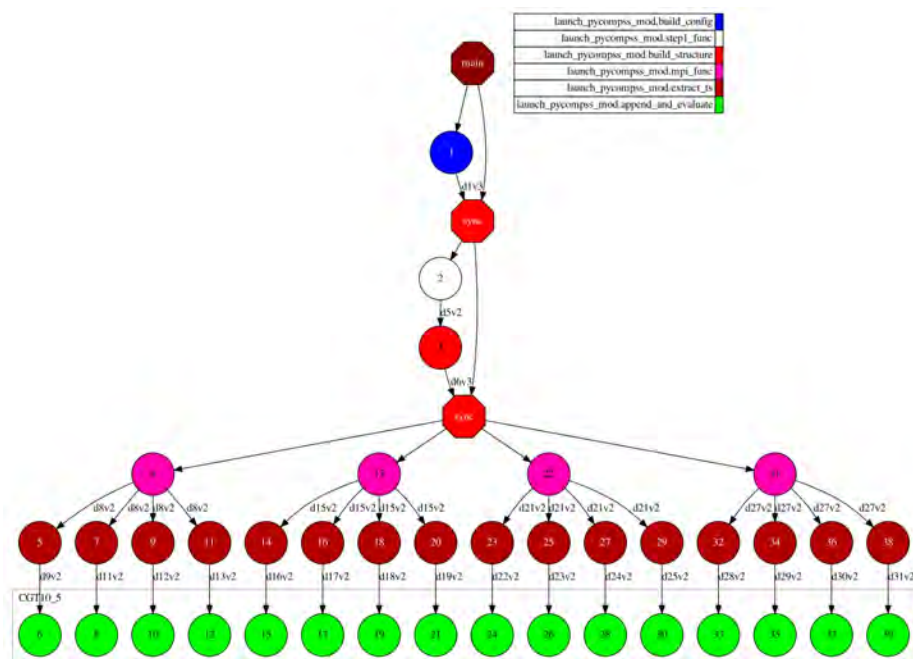


Figure 11. PyCOMPss graph of a full PTF workflow run (for a small example of 16 scenarios) with all the tasks and dependencies.

## 1) Ensemble manager (task 1-2)

The ensemble manager has been developed as a module (task1 see Figure 10) that takes as input short term information on the earthquake from the seismic monitoring (first estimations of Magnitude and Hypocenter and relative uncertainty) as well as long term information (mainly based on historical seismicity, fault catalog, and other long-term hazard components) to generate a list of scenarios defined by sets of 10 parameters corresponding to the description of an Okada fault model, the model used by the simulation code HySEA to generate and propagate tsunamis (Selva et al. 2021, [S1]). From this list, an ensemble of scenarios is sampled to estimate source uncertainty (Cordrie et al. 2022 [ens1]). The user can decide on the size of the scenario ensemble and on the sampling method (Classic Monte Carlo or Latin Hypercube) and the ensemble manager will write each set of parameters (scenarios) in a file ready to be used as input by HySEA.

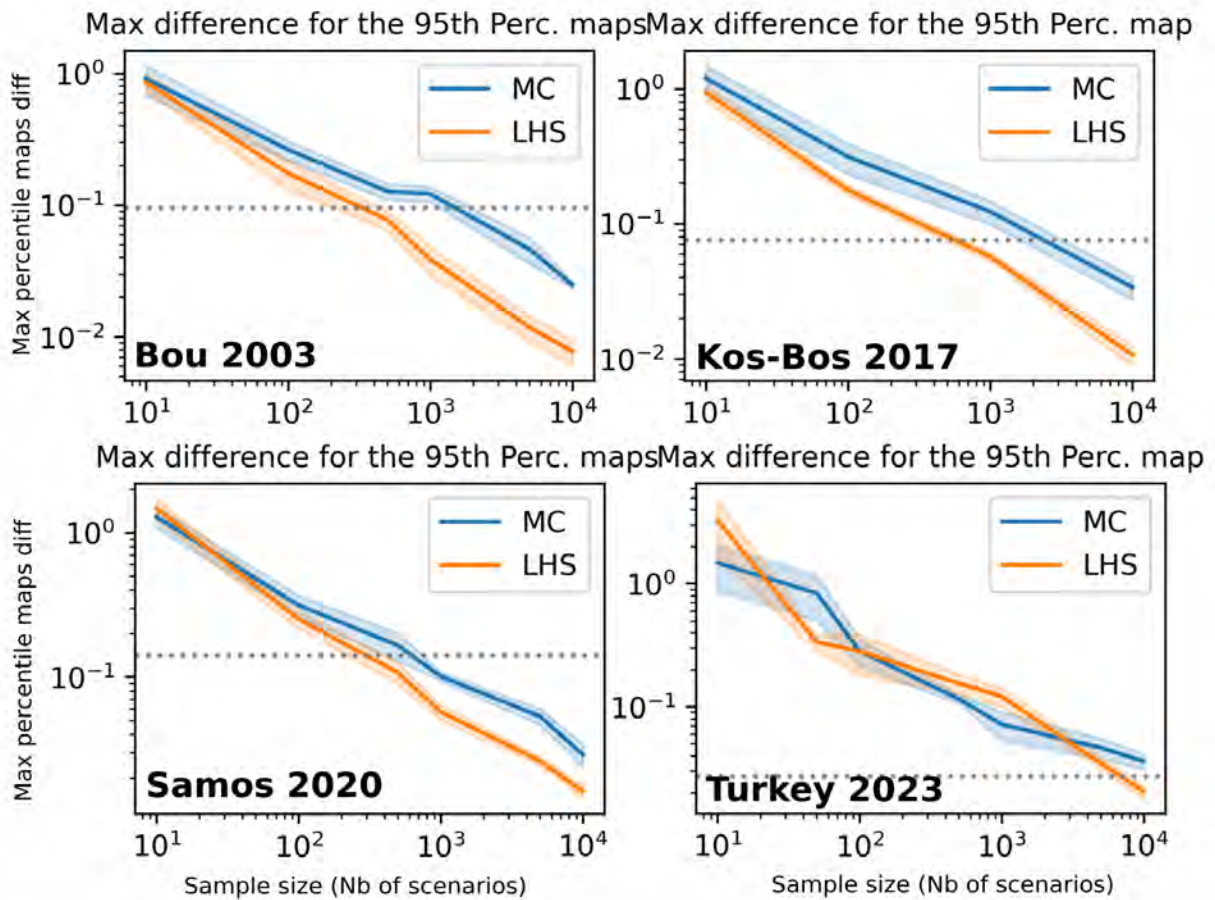


Figure 12. Convergence of the PTF results for multiple Monte-Carlo sampled ensembles.

## 2) HySEA Simulations (task 3-4)

Once the ensemble of scenarios to be simulated is received, the task orchestrated by PyCOMPSs takes charge of grouping the scenarios into different chunks, adjusting their sizes based on the cluster's capacity and the number of available GPUs. This ensures that a specific number of simulations are processed within a single job, while multiple jobs run in parallel to optimize the entire process. The creation of parameter files required for each simulation and their distribution within the file system is automatically handled by PyCOMPSs. In each of these jobs, the Monte Carlo version of the Tsunami-HySEA code is executed, enabling the simultaneous launch of a predetermined number of scenarios, with each assigned to a GPU. This efficient orchestration ensures traceability and allows for the seamless execution of multiple simulations with improved performance, see Figure 12.

## 3) Intermediate Forecast delivery and convergence monitoring (task 5-7)

The multitask system of PyCOMPSs allows the simultaneous realization of multiple tasks and consequently brings the possibility of interacting with the ongoing parallel tsunami simulations. The workflow integrates the possibility of progressively retrieving the results of simulation, producing early stage forecasts and monitoring the convergence toward stable results without waiting for all the simulations to complete. The user can decide the frequency of the delivery in



terms of the number of available simulations. This is also made possible with several post-processing steps which adapt automatically to the number of available simulation outputs.

The output of HySEA is post-processed by two scripts called at different times of the workflow. The first one is called right after the end of each simulation, it extracts from the NetCDF HySEA output a set of predefined values. The second one is called when new results (early stage or final) are produced, that is at each predefined interval (number of ended simulations). This script merges the available scenario's post-processed information into a unique NetCDF file that is used as input for the aggregation step. For these two scripts, two versions are available. One is based on python scripts, which generate for each simulation temporary NetCDF files. The other version is based on Ophidia, which keeps in memory all the results and produces the final NetCDF when requested (see Section 3.2.3), avoiding the multiple file creation and reading of the python scripts.

A convergence module was also introduced in the intermediate steps to follow the evolution of the accuracy and precision of the forecast generated by these ensembles of simulation outputs produced at early stages (Example of the convergence module outputs for a delivery rate of 10 scenarios in Figure 13).

To improve event diagnostics, in this phase tsunami simulation execution is checked. In case of simulation failures (due to extreme parameter's value, memory issues or else), a function reports the failed scenario's identification number into a text file which can be checked by the user and that is passed in input of the aggregation step which will ignore these scenarios in the following forecast calculations, producing consistent results.

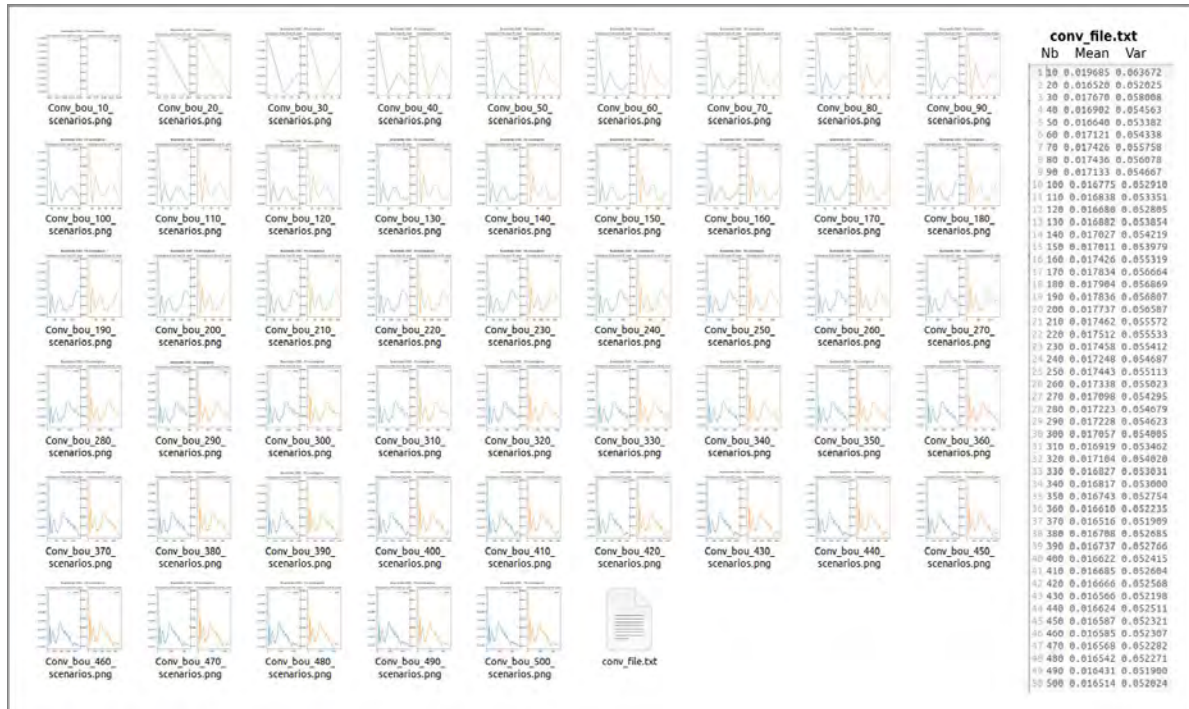


Figure 13. Files produced progressively by the convergence module at each delivery step (every 10 scenarios for a run of 500 scenarios) during the workflow to allow the user to follow the convergence of the results. The plots correspond to the mean and variance of the PTF.



#### 4) Integration of incoming data and update of the PTF (task6)

The first scenario ensemble is generated based on the first estimates of the magnitude and epicenter provided generally a few minutes after an earthquake. However, rapidly new information becomes available, such as the focal mechanism and, later on, the first tsunami observations, e.g. from tide-gauges or offshore buoys. This information may be used to update the PTF, improving the accuracy and hopefully increasing the precision (decrease of uncertainty) of forecasts with better constrained input information (Cordrie et al. 2023a,b [ens1,ens2]).

To this end, just before the aggregation step, two python modules of probability update can be called if the corresponding options are activated. The modules take as input incoming information about the seismic source or the tsunami to weight each scenario by re-evaluating their probabilities with regards to this new information. Since these modules impact the list of probabilities (output of the ensemble manager) but not the simulations themselves they can be called in parallel to the simulations and also at intermediate steps.

The module `run_step_kagan.py` takes in input the three angles (Strike,Dip, Rake) from the focal mechanism solution of the target event and compute for each scenario a 3D angular distance (Kagan angle) between this “observed” plane and the plane orientation (strike, dip and rake angles) of the scenario. From this value, a weight is attributed to each scenario using an exponential probability density function. Each scenario’s probability is then used to weight the scenario results during the aggregation phase, producing new forecasts based on updated information on the source.

Similarly the module `run_step_mare.py` takes in input the maximum wave height measured at specific measure points (e.g. tide-gages). The input consists of a list of maximal wave amplitudes at different locations. The script computes, for each scenario, a Normal Root Mean Square misfit value between the distribution of the observation and the distribution of the simulation wave heights. From this value a weight is attributed to the scenario’s probability using an exponential probability density function. Scenario weights are then used to update the PTF forecast during the aggregation phase.

The two updates can be superimposed and a future objective would be to add data listeners to these modules in order to perform automatic updates of the PTF depending on the available information at each PTF delivery time.

An example of the results obtained with these update methods is shown in Figure 14. It is a PTF run of the 2023 Turkish earthquake, where the update of the PTF using the tsunami data improves the results by providing forecasts closer to the actual tsunami observations.

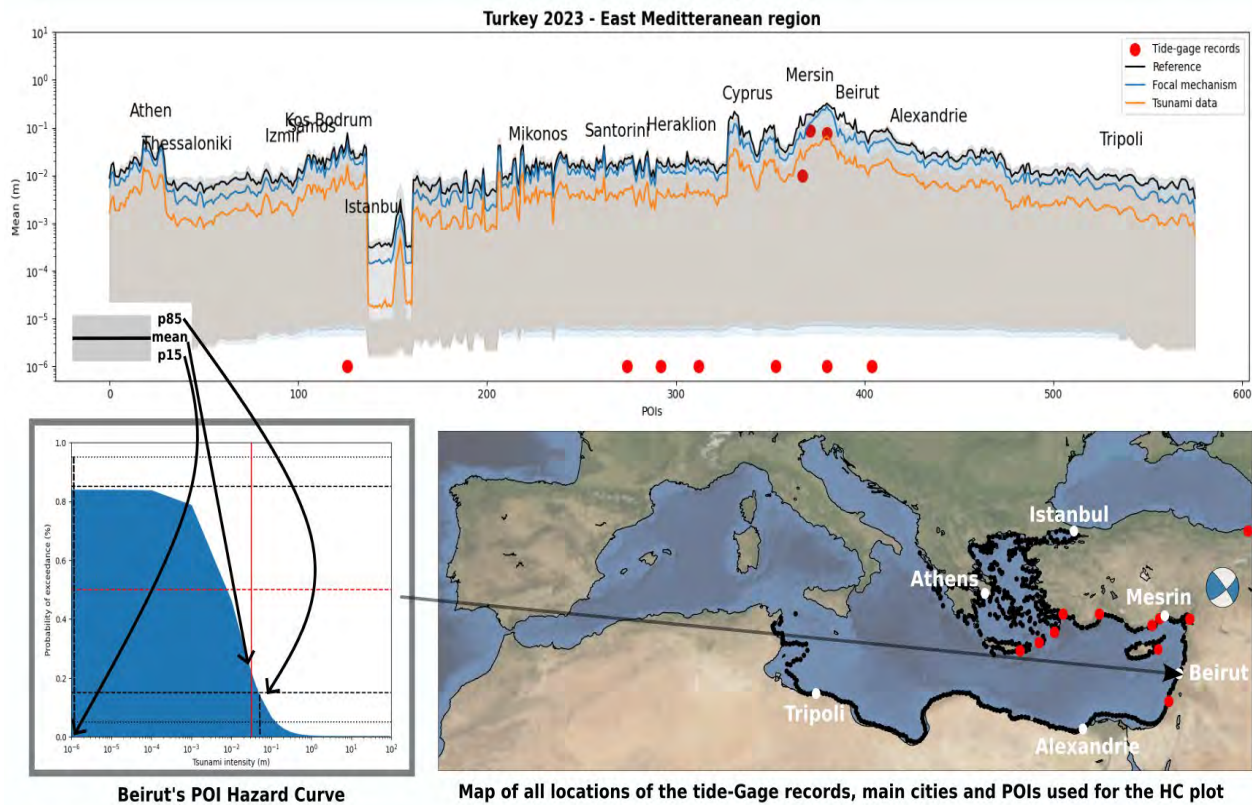


Figure 14. Example of PTF results (Hazard Curve and extracted percentiles and mean values) obtained by the classic calculations (black line), by the update using the Focal Mechanism (blue line) and by the update using the Tsunami Data (orange line) for the 2023 Turkey earthquake.

### 3.2.2. Deployment with HPCWaaS

The eFlows4HPC HPCWaaS (HPC Workflows as a Service) platform is designed to facilitate the deployment and reusability of complex workflows on federated HPC infrastructure. It is within the three use case Pillars of the eFlows4HPC project that the HPCWaaS platform and eFlows4HPC software stack will be evaluated. The HPCWaaS platform supports the roles of both workflow developer (through the Development Interface<sup>2</sup>,) and the end-user (through the Execution API<sup>3</sup>).

The Alien4Cloud platform provides the developer with a versatile tool for managing applications for cloud computing. Figure 15 displays the user interfaces for selecting or beginning applications, and for editing the topology associated with a given application. Figure 16 displays a segment of the TOSCA description for the PTF workflow that is generated by and exported from Alien4Cloud.

<sup>2</sup>[https://eflows4hpc.readthedocs.io/en/latest/Sections/03\\_HPCWaaS\\_Methodology/01\\_Development\\_Interface.html](https://eflows4hpc.readthedocs.io/en/latest/Sections/03_HPCWaaS_Methodology/01_Development_Interface.html)

<sup>3</sup> [https://eflows4hpc.readthedocs.io/en/latest/Sections/03\\_HPCWaaS\\_Methodology/02\\_Execution\\_API.html](https://eflows4hpc.readthedocs.io/en/latest/Sections/03_HPCWaaS_Methodology/02_Execution_API.html)

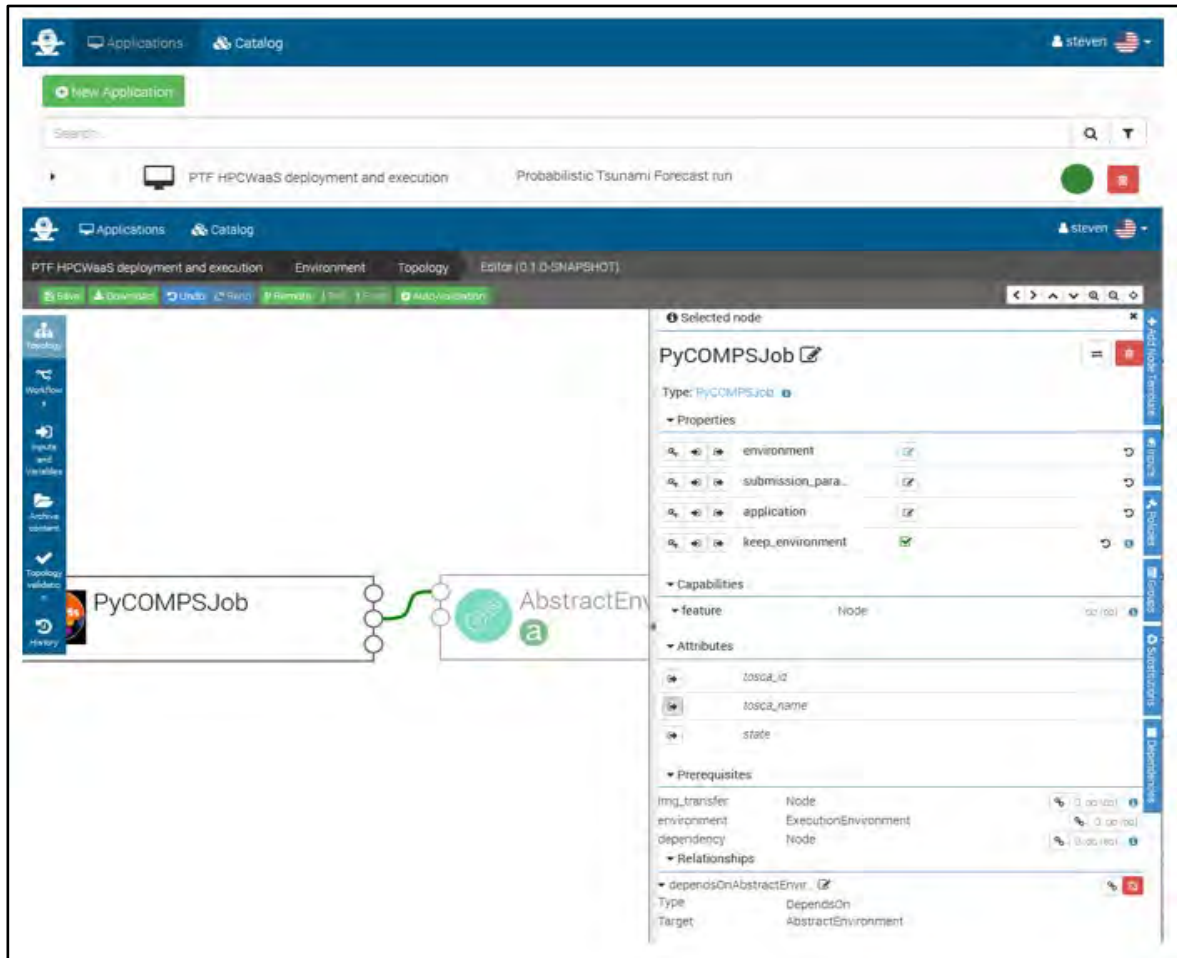


Figure 15. Application menu and topology editing functionality with Alien4Cloud.

```
tosca_definitions_version: alien_dsl_3_0_0

metadata:
  template_name: PtfTest2
  template_version: 0.1.0-SNAPSHOT
  template_author: steven

description: ""

imports:
  - eflows4hpc.env:1.1.0
  - yorc-types:1.1.0
  - toska-normative-types:1.0.0-ALIEN20
  - alien-base-types:3.0.0
  - org.eflows4hpc.pycompss.plugin:1.1.0

Topology template:
  node_templates:
    AbstractEnvironment:
      metadata:
        ahc_edit_x: 220
        ahc_edit_y: 0
      type: eflows4hpc.env.nodes.AbstractEnvironment
    PyCOMPSJob:
      metadata:
        ahc_edit_x: 0
        ahc_edit_y: 0
      type: org.eflows4hpc.pycompss.plugin.nodes.PyCOMPSJob
      properties:
        submission_params:
          qos: debug
          python_interpreter: python3
          num_nodes: 1
          extra_compss_opts: "--env_script=${app_path}/env.sh --exec_time=${exec_time} --worker_working_dir=${run_path}"
          application:
            container_opts:
              container_opts: ""
            arguments:
              - "--seis_type ${seis_type}"
              - "--parameters_file ${parameters_file}"
              - "--data_path ${data_path}"
              - "--run_path ${run_path}"
              - "--templates_path ${templates_path}"
              - "--kagan_weight ${kagan_weight}"
              - "--mare_weight ${mare_weight}"
              - "--hours ${hours}"
              - "--group_sims ${group_sims}"
              - "--cfg from template"
              - "--event ${event}"
            command: "${app_path}/Code/launch_pycompss.py"
            keep_environment: true

      requirements:
        - dependsOnAbstractEnvironmentExec_env:
            type: requirement: environment
            node: AbstractEnvironment
            capability: eflows4hpc.env.capabilities.ExecutionEnvironment
            relationship: toska.relationships.DependsOn

      workflows:
        PtfExecution:
          inputs:
            user_id:
              type: string
              required: true
              vault_id:
                type: string
                required: true
            app_path:
              type: string
              required: true
            run_path:
              type: string
              required: true
            data_path:
              type: string
              required: true
            templates_path:
              type: string
              required: true
            parameters_file:
              type: string
              required: true
            num_nodes:
              type: integer
              required: false
              default: 1
            exec_time:
              type: integer
              required: false
              default: 20
            event:
              type: string
              required: true
            seis_type:
              type: string
              required: false
              default: BS
```

Figure 16. Excerpt from the TOSCA description of the initial PTF Workflow.

Alien4Cloud also facilitates execution of the workflow with the prescribed parameters following specification of the topology. Figure 17 displays selection of the chosen HPC resource and Figure 18 displays specification of execution and submission parameters. With the addition of appropriate authentication credentials, deployment can be triggered within Alien4Cloud.

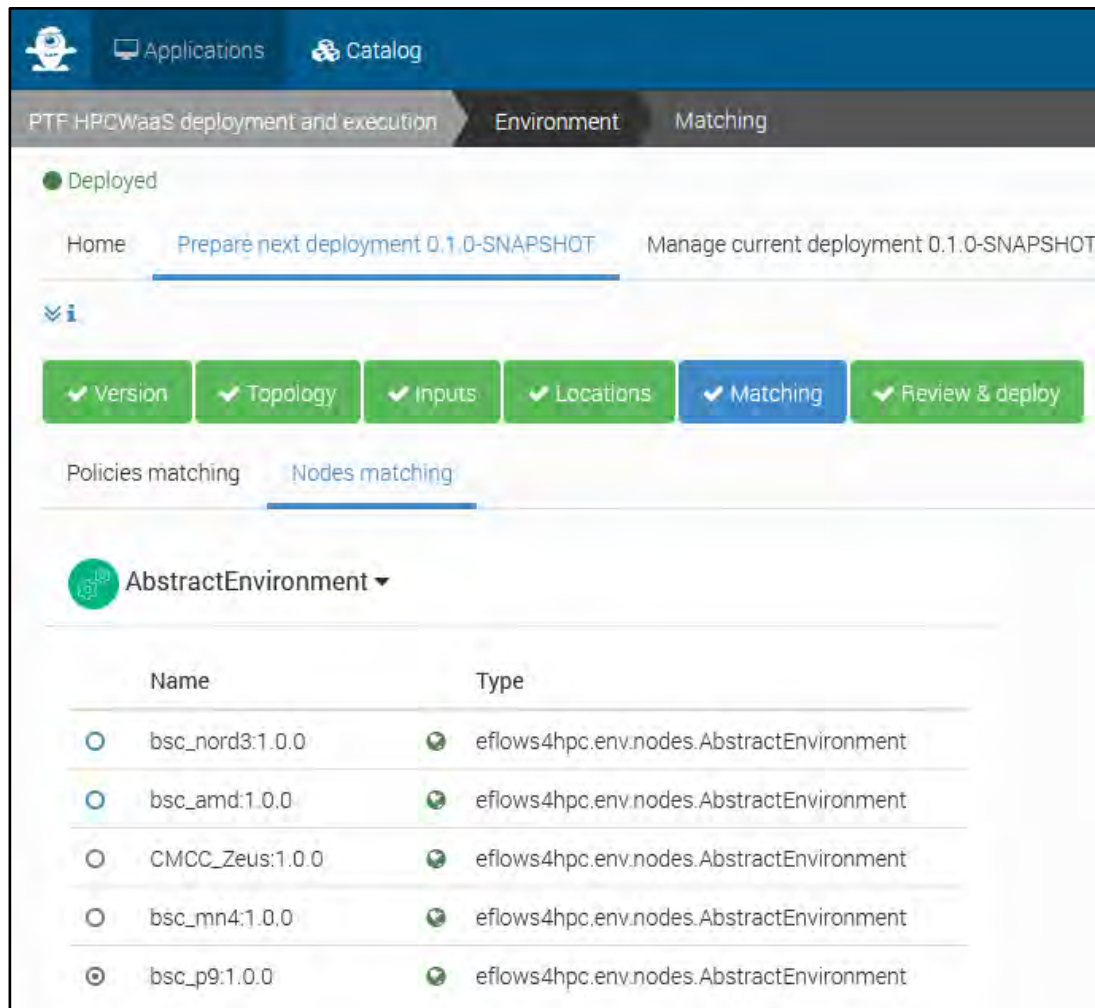


Figure 17. Menu of federated HPC facilities on which the workflow could be deployed.

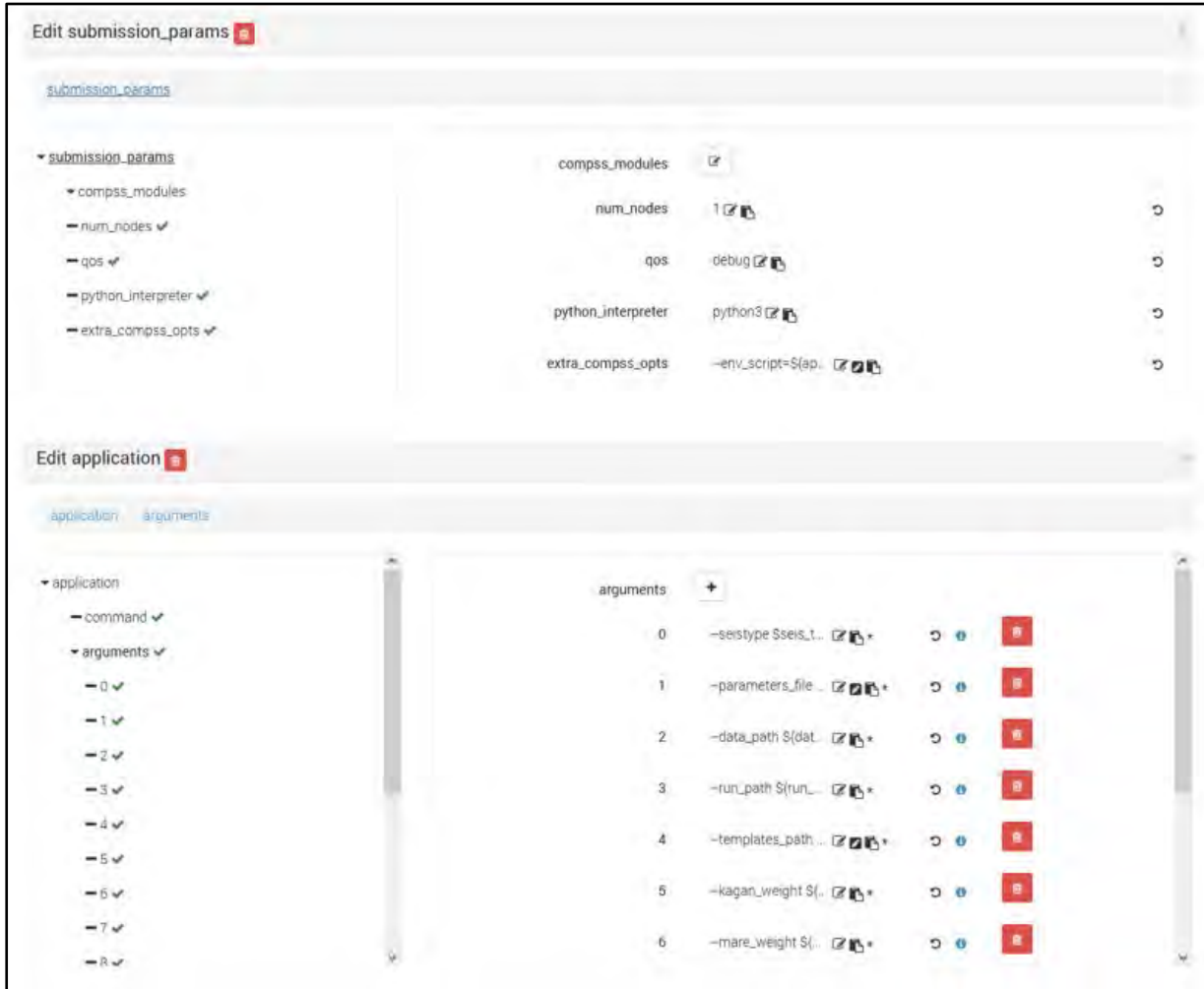
### On-going work:

The input data will be stored into a registry (B2DROP or B2SHARE). The output data will be stored to another repository (B2DROP or B2SHARE).

The TOSCA description should be implemented with new data pipelines (ROM workflow) to generate automatic data transfer operation during the workflow for the intermediate PTF evaluations and at the end.

The workflow will be fully containerized. It is in development.





The screenshot displays two main configuration panels in the Alien4Cloud interface:

- Edit submission\_params:** This panel is divided into two sections. The left section shows a tree view with expandable items: `submission_params`, `compss_modules`, `num_nodes`, `qos`, `python_interpreter`, and `extra_compss_opts`. The right section shows the selected values for these parameters:
  - `compss_modules`: (empty)
  - `num_nodes`: 1
  - `qos`: debug
  - `python_interpreter`: python3
  - `extra_compss_opts`: `-env_script=$($ap...`
- Edit application:** This panel also has two sections. The left section shows a tree view with `application` and `arguments`. The right section shows a list of arguments indexed from 0 to 6:
  - 0: `--seis_type $seis_t...`
  - 1: `--parameters_file ...`
  - 2: `--data_path $dat...`
  - 3: `--run_path $run...`
  - 4: `--templates_path ...`
  - 5: `--kagan_weight $...`
  - 6: `--mare_weight $...`

Figure 18. Specification of submission parameters and arguments for deployment of workflow on the selected HPC resource within Alien4Cloud.

### 3.2.3. Data analytics with Ophidia

The processing of the simulation results (point 3 in Section 3.2.1) using python scripts requires two steps: a first script producing an individual post-processed *.nc* (NetCDF) file at the end of each simulation and a second script reading (at regular intervals, every Nth scenario) all the individual files to produce a single *.nc* file to be provided as input to the PTF aggregation and the hazard curves calculation. The writing of the intermediate individual NetCDF files is a computationally heavy task both in terms of time and memory, particularly in the cases with hundreds to thousands of scenarios.

The Ophidia framework incorporated in the workflow, see Figure 19, (<https://www.cmcc.it/data-services-and-products/software-lists/ophidia>) enables immediate post-processing operations for each of the time series output from the tsunami simulations, storing them into datacubes and merging them into the final *.nc* file, without creating the intermediate individual files. The post-processing operations are related to the wave amplitude variable calculation: the maximum, the minimum, the peak-to-trough, the Green's amplification and the removal of the offset with respect to the sea level before the tsunami, if needed.

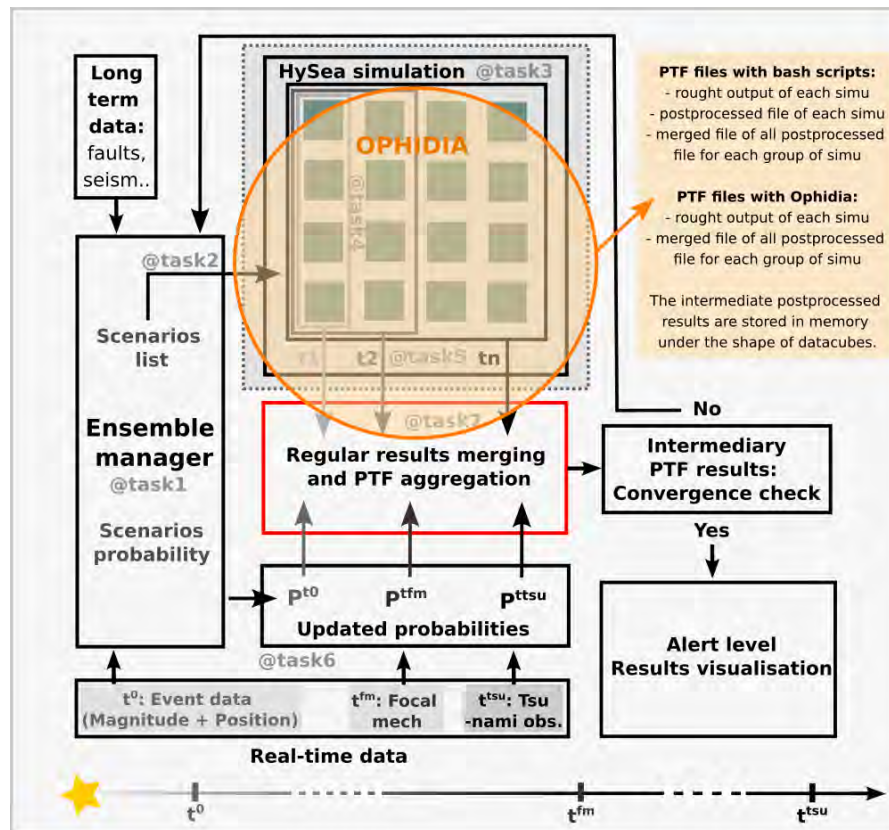


Figure 19. Schematic diagram of the PTF workflow with the indication on where and how Ophidia can improve the post-processing tasks.

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 original 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, when a scenario file is available, some post-processing operations are performed in-memory into Ophidia datacubes which are merged to the final resulting datacube (one for each computed variable). This process is computed until the number of scenarios reaches a certain value (e.g. 20 scenarios) and a final NetCDF file containing all the resulting variables is saved on disk, see Figure 20.

The main advancements with respect to the first release involve:

- the use of the `to_dataset()` operation useful for converting an Ophidia datacube into an Xarray dataset before saving to disk;
- the `oph_mergecubes2` operator extended to support the aggregation of two cubes with different structure; specifically, this is useful to progressively create the final cube by appending each time the different scenarios' values.
- the integration of the Ophidia-PyCOMPSs that improves HPC resources usage through the prolog-epilog mechanism to support the activation/deactivation of the Ophidia internal services (as fully described in the Deliverable D2.4);



- the porting and testing of this sub-workflow on Power9 machine.

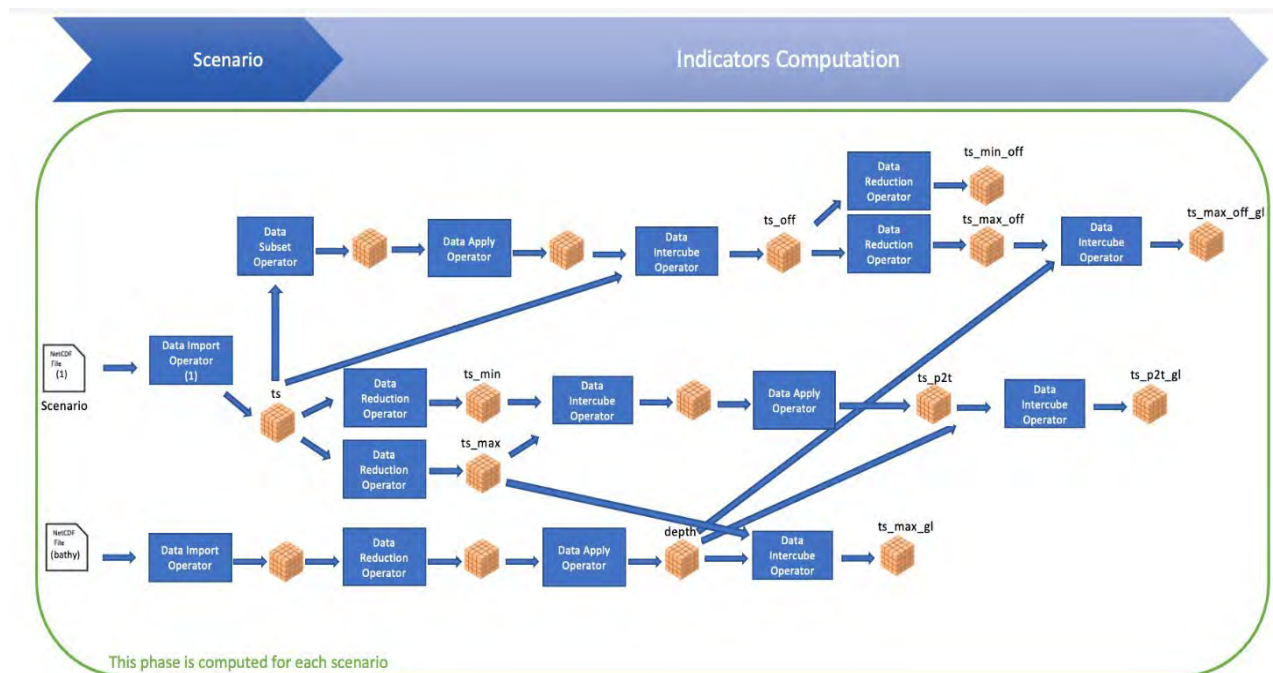


Figure 20. Internal steps for the computation of the variables for each scenario.

## 4. Exploratory new components for the workflows

This section details a number of research developments into new components for the UCIS4EQ and PTF workflows that are not yet at a Technology Readiness Level for inclusion in the operational workflows, but that have exposed important considerations regarding what is needed for their safe implementation. Both earthquake and tsunami workflows use computationally intensive physics simulations to calculate the outcomes of multiple scenarios. The components discussed here are, in general, emulators which seek to use machine learning (ML) to predict the output of the physics simulations based upon an appropriate training set. The primary motivation for emulators is to reduce the time-to-solution. The primary challenge is to have sufficient confidence in the accuracy of the emulator outcomes.

Both workflows address emergency situations, most likely with new scenarios that may not have been considered previously. The ultimate inclusion of the emulator processes in the operational workflows will depend upon our ability to mitigate any disadvantages this may bring. One possibility is that we perform lower resolution physics simulations, or physics simulations that calculate a partial solution, with an ML model that is able to calculate the full-resolution outcome based upon this starting point. Another possibility is that the workflow can resort to using exclusively the physics simulations in the absence of an appropriate ML model, or divide the set of scenarios into those that can and those that cannot be computed with the aid of an emulator. In any case, the emulator must demonstrate a significant time-to-solution advantage over the full physics simulation in order to justify its inclusion.

## 4.1. MLEsMap workflow with dislib and PyCOMPSs

The Machine Learning based Estimator for ground Shaking maps (MLEsMap) is a new tool built in the framework of the eFlows4HPC project aiming to provide ground acceleration in a given region a few seconds after a large magnitude earthquake occurs. MLEsMap uses the ML engine in conjunction with big data catalogs of physics-based seismic simulations (Monterrubio et al., 2023b [S11], 2023c [S12]).

In Pillar III, MLEsMap has been proposed to provide offline-trained models for an online service defined in the UCIS4EQ workflow, specifically in the so-called Building Block 8 (see D6.1).

For modeling, MLEsMap uses the Random Forest regressor provided in the dislib software, and Deep Neural Networks from the TensorFlow python library. It is worth mentioning that the Random Forest regressor algorithm was developed, tested, and included in the dislib library by the WP1 team as a requirement collected in Phase 1.

The development of MLEsMap has been split between two phases along the project. As was reported in the D6.3 UCIS4EQ activity 4, in the first phase (M6-M18) a prototype of the MLEsMap has been designed and implemented using one of the largest data sets of simulated ground motion data with more than half a million hypothetical earthquakes. This dataset, namely CyberShake Study 15.4, is a physics-based Probabilistic Seismic Hazard model for Southern California at 1 Hz, provided in collaboration with the Southern California Earthquake Center, SCEC<sup>4</sup>.

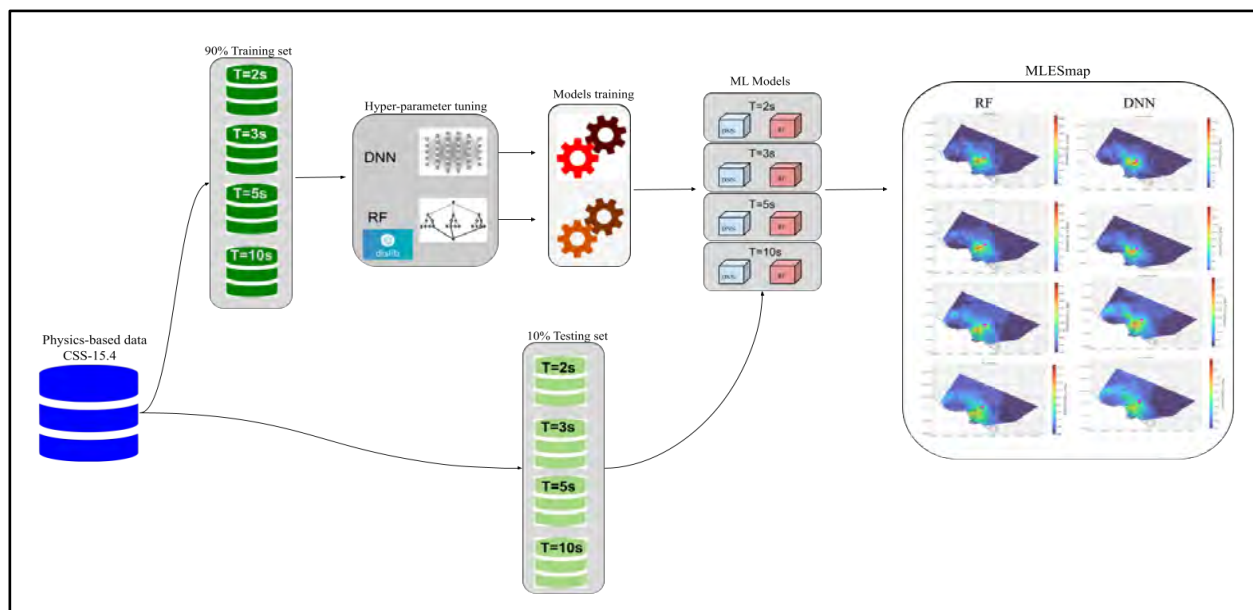


Figure 21. MLEsMap methodology for the prototype developed at Iteration 1 - Phase 2 (M6-M18).

MLEsMap is in itself a methodology that includes the workflow presented here. The MLEsMap methodology (see Figure 21) uses disaggregated frequency components of the database and trains an ML model that can be used to produce shakemaps in a matter of seconds, compared to the hours required to fulfill a similarly accurate 3D physical simulation.

<sup>4</sup> [https://strike.scec.org/scecpedia/CyberShake\\_Study\\_15.4](https://strike.scec.org/scecpedia/CyberShake_Study_15.4)

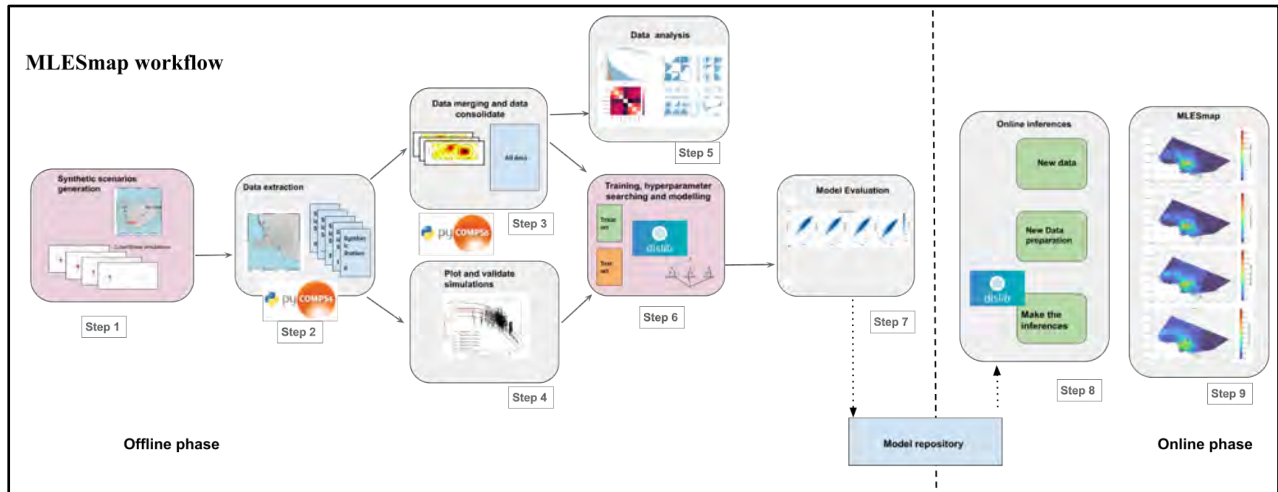


Figure 22. MLEsmap workflow developed at iteration 3 phase 2 (M19-M30). Pink boxes show the steps that must be executed in a HPC environment.

The MLEsmap workflow (see Figure 22) directly uses the synthetic database to fulfill the model training, using PyCOMPSs to leverage the data management steps. Specifically, in the third phase of the project (M19-M30), it was proposed to generalize the MLEsmap methodology to obtain ML models for regions other than California. The first implementation of the MLEsmap workflow took as a study case the Southern Iceland Seismic Zone (SISZ) and the Reykjanes Peninsula Oblique Rift (RPOR) region.

The MLEsmap workflow in Figure 22 defines 9 steps:

#### Step 1 Synthetic scenarios generation:

The generation of the dataset is the most computationally expensive step of the MLEsmap workflow. For this purpose we use the open-source CyberShake platform developed to simulate ground motion in specified sites using forward physics-based simulations for modeling large amounts of hypothetical earthquakes in California. The CyberShake platform is a complex workflow by itself, that uses pre-existing codes as components. Thanks to the ChESEE project, the CyberShake platform was installed at the MareNostrum supercomputer in collaboration with SCEC. The goal of that effort was producing probabilistic seismic hazard (PSHA) maps of a region in Iceland, by combining a large set of hypothetical earthquakes (Rojas et al. 2021, [S10]). As a byproduct, the individual realizations of the earthquakes can now be used to feed the MLEsmap methodology. Unfortunately, the first batch of simulations were too few and we have, within the eFlows4HPC project, enabled a higher resolution run, specifically for its use for MLEsmap. In particular the simulations focused on the Southern Iceland region. As a workflow manager we employed UnifiedCSFlow, developed at BSC<sup>5</sup>. CS executions scale with the number of stations where the ground motion proxies are saved. The computational resources per station of the different CyberShake stages are shown in Table 2.

<sup>5</sup> <https://zenodo.org/record/6382176>

Table 2. Computer resources per each station in a Cybershake execution for the Iceland use case.

CS Stages	CPUs	Nodes	Tasks	Runtime
pre-SGT	48	1	1	1 min.
pre-AWP	48	1	1	15 sec.
AWP_X	576	12	576	20 min.
AWP_Y	576	12	576	20 min.
post-X	48	1	1	10 min.
post-Y	48	1	1	10 min.
run DS	576	12	288	5 min

To generate the dataset needed for the South Iceland region, we obtained a grant by the National Supercomputing Network (RES) AECT-2022-2-0025 with the title “Machine-Learning based Estimator for ground motion Shaking maps (MLESmap)” that was awarded with a total of 610,000 cpu-hours. Our dataset contains the synthetic waveforms recorded in 593 stations and generated due to 6000 hypothetical earthquakes.

### Step 2 Data Extraction

The data extraction is the step where the raw data coming from Step 1 must be accessed, processed, and extracted to obtain the desired ground shake intensity measures such as peak spectral acceleration (PSA), peak ground acceleration (PGA), peak ground velocity (PGV), among others. These are the quantities of interest of MLESmap. This step must be done per each station independently. To optimize the time to complete this step PyCOMPSs has been integrated on top of it to manage the task at a station level improving the parallelism required.

### Step 3 Merging and data consolidate

Once the step 2 is finalized the Step 3 retrieves all the information at each station and merges it to generate and to consolidate the dataset that contains the whole information of the simulation parameters per each synthetic earthquake and its associated intensity measures. To improve the access to the data this Step is also orchestrated by PyCOMPSs.

### Step 4 Plot and validate simulations

This step is a quality control step that visualizes, analyzes and compares the simulation results with those computed with Ground Motion Models (GMM). In the future we plan to join Step 3 and Step 4 with a single PyCOMPSs task.

### Step 5 Data Analysis

The Data Analysis step addresses the analysis of the consolidated data to analyze its statistical characteristics, which can become fundamental towards parameterising the training process.

### **Step 6 Training, hyperparameter searching and modelling**

Step 6 requires an HPC environment to be executed in a reasonable time. This step relies on the *dislib* library to train the ML models using a random forest regressor.

### **Step 7 Model evaluation**

The model evaluation is done using the test set and indicates the quality of the inferences provided by the MLEsmap. Hence this is a key step in order to quantify the predictive power of the models obtained. If the evaluation is positive, the qualified models can be stored in a model repository that will be used for an online application, such as UCIS4EQ.

### **Step 8 On-line inferences**

Inferences are performed from primary earthquake information (magnitude and location). Query times are in the order of a second, which makes real-time assessment a possibility.

### **Step 9 Map plots**

Lastly, the MLEsmap plots are generated to provide actionable results, for a given earthquake and region.

Our approach (i.e. simulate, train, deploy) can help produce the next generation of ground shake maps, capturing physical information from wave propagation (directivity, topography, site effects) at the velocity of simple empirical GMPEs.

The MLEsmap code accessibility is described at section 5 of this document.

## **4.2. Emulators for Tsunami propagation**

### **4.2.1. Tsunami inundation: Considerations for Training Sets**

The existing version of the PTF calculates maximum inundation heights predicted from simulation results offshore. This allows the tsunami impact at many different coastal locations to be estimated for a given scenario using a single numerical simulation. If we want to calculate the inundation at high spatial resolution for one or more coastal regions, then we need to solve the Nonlinear Shallow Water Equations on a nested or telescopic grid at each site. Each successive grid level consists of more numerous and smaller cells, demanding more numerous and smaller time-steps. The computational effort and time-to-solution of a complete inundation calculation can be orders of magnitude greater than that for the coarsest grid alone. Although the grid of highest resolution may consist of several millions of points (a 5 or 10 meter resolution is typical), the dimensionality of the set of possible outcomes will be far lower. For example, some locations at high elevation will never be inundated. There will also be a high correlation between the inundation at neighboring locations such that there is significant redundancy in the data points representing the inundation maps.

In the same way that a Singular Value Decomposition can represent vast linear relationships using relatively few parameters, we can hypothesize that an encoder-decoder model (Figure 23) would be able to calculate a low-dimensional parameter space with which we can predict any outcome from our model space. The model (Figure 23) finds the most appropriate set of coefficients for the latent space from the input simulation grid using the encoder. We then reconstruct the grid using



the decoder; the accuracy of the model can be evaluated by calculating the residuals between the input and reconstructed inundation maps. In our application, we would want the encoder to act on input offshore time-series and the decoder to calculate the corresponding emulated inundation maps (Figure 23).

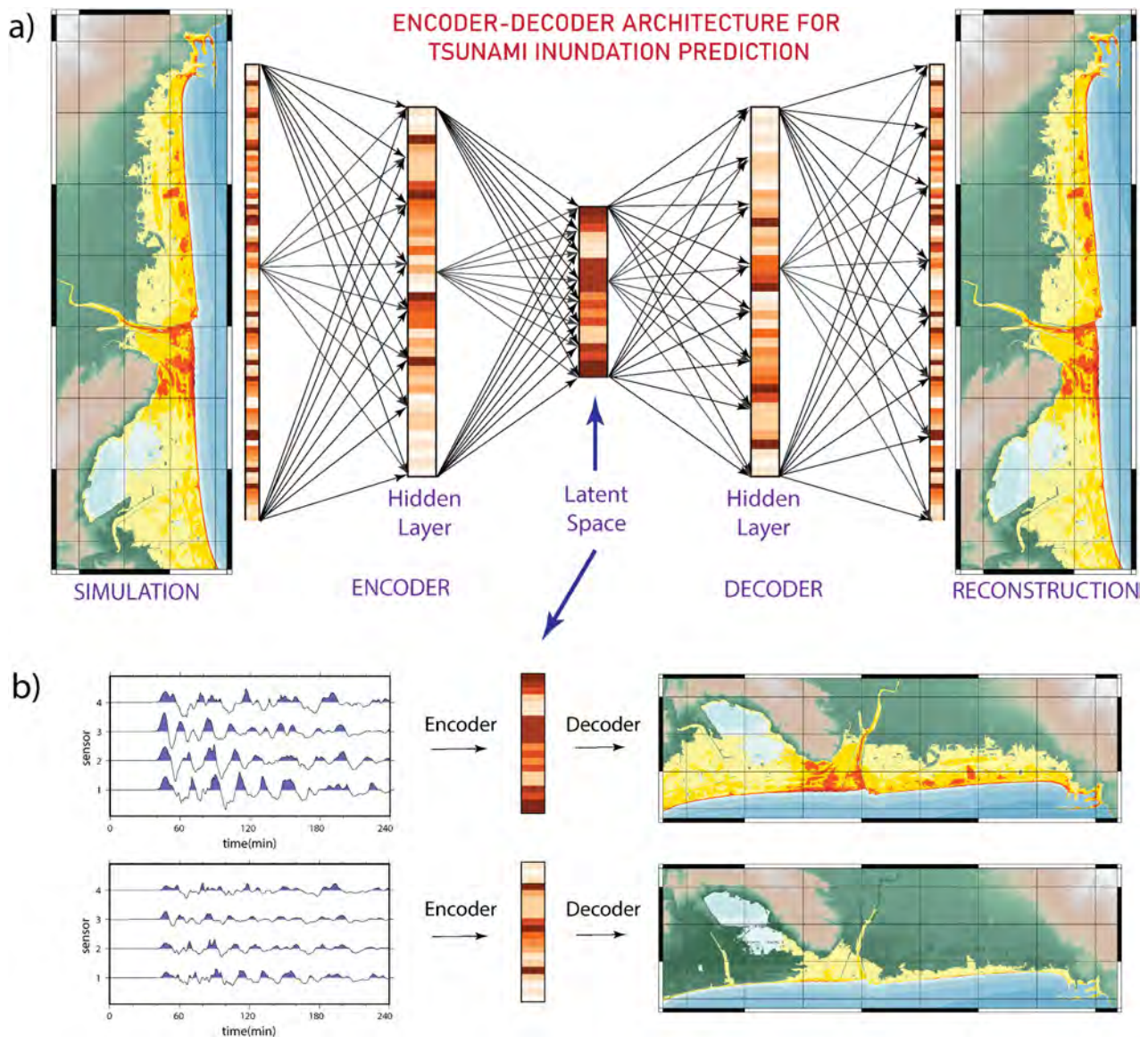


Figure 23. Sketch of an encoder-decoder architecture for predicting high-resolution maps of maximum onshore tsunami inundation from the (far cheaper computationally) offshore time-series. The justification for the model is that both the offshore time-series and the inundation maps demand vast dimensions of data points, but that the fundamental forms of both can be specified by a far smaller parameter space. The encoder and decoder are trained simultaneously. Panel (a) displays how we could model a low-dimensional parametrization of our full set of inundation simulation results such that any input inundation map can be adequately modeled with a given specification of our latent space. Panel (b) displays the encoder-decoder which models the parameters for our latent space from input time-series and which outputs a reconstructed inundation map.

We evaluated the possibilities and the limitations of such an emulator using the inundation simulations calculated during the ChESEE project [S2] in a Probabilistic Tsunami Hazard Analysis



for the eastern coast of Sicily [S3]. One of our main concerns in applying an ML model in an emergency computing situation is that the most extreme cases of inundation, those of most concern for society, are poorly represented in the training set given that they belong to the lower probability, higher impact, tail of the probability distribution. To demonstrate, we display in Figure 24 the proportion of scenarios from the PTHA study that result in the indicated level of inundation at different locations in the Bay of Catania. We see immediately that large regions of the map are either never inundated for the available set of training events, or are inundated for a tiny proportion (e.g. less than 0.1%) of the scenarios. A physics simulation should be able to model the outcome of any realistic input scenario. An ML model on the other hand is likely to be severely limited to its training set, unable to extrapolate beyond the most extreme inundations well represented in the training set. This is especially likely given the non-linearity in the physics simulations.

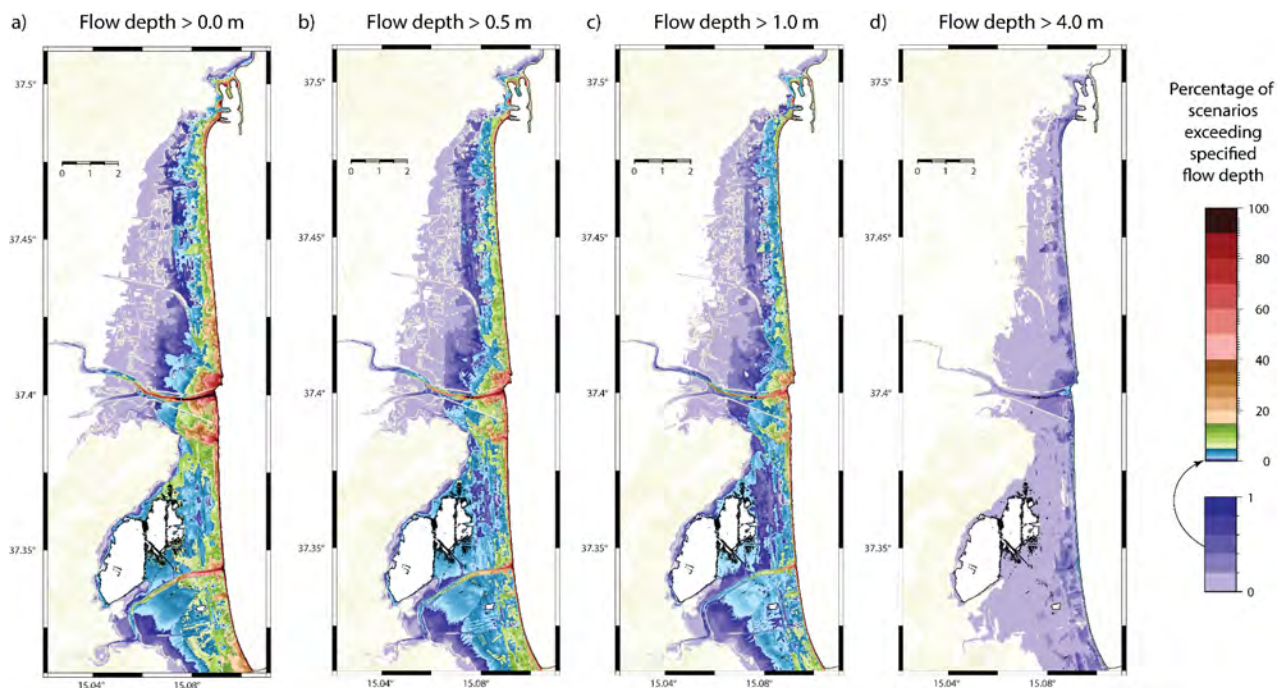


Figure 24. The percentage of earthquake tsunami scenarios in a PTHA study [S3] exceeding the flow depth indicated across the highest resolution inundation grid for the Bay of Catania, Sicily. There are a total of 32363 scenarios considered in this figure and so any location colored to indicate fewer than 1% of the scenarios has fewer than 323 scenarios in the dataset exceeding the specified inundation. Panels a), b), c), and d) indicate flow depths exceeding 0m, 0.5m, 1m, and 4m respectively.

We demonstrate this in practice for a preliminary training set in which the training scenarios are picked essentially at random from the total set of scenarios. Figure 25 displays how the locations closest to the sea front (that are inundated in essentially all of the scenarios) are predicted fairly well, at least statistically, by the encoder-decoder model. The locations that are further inland are, in most cases, inundated in far fewer of the training scenarios. The pattern is not clear cut as of course the outcome depends not only upon the distance from the waterfront but also on the elevation and on the “path” to the location (i.e. the elevation at all points on the potential routes that the incoming waves can take to reach that point). We conclude that more scenarios representing more extreme cases of inundation are needed in the training data.

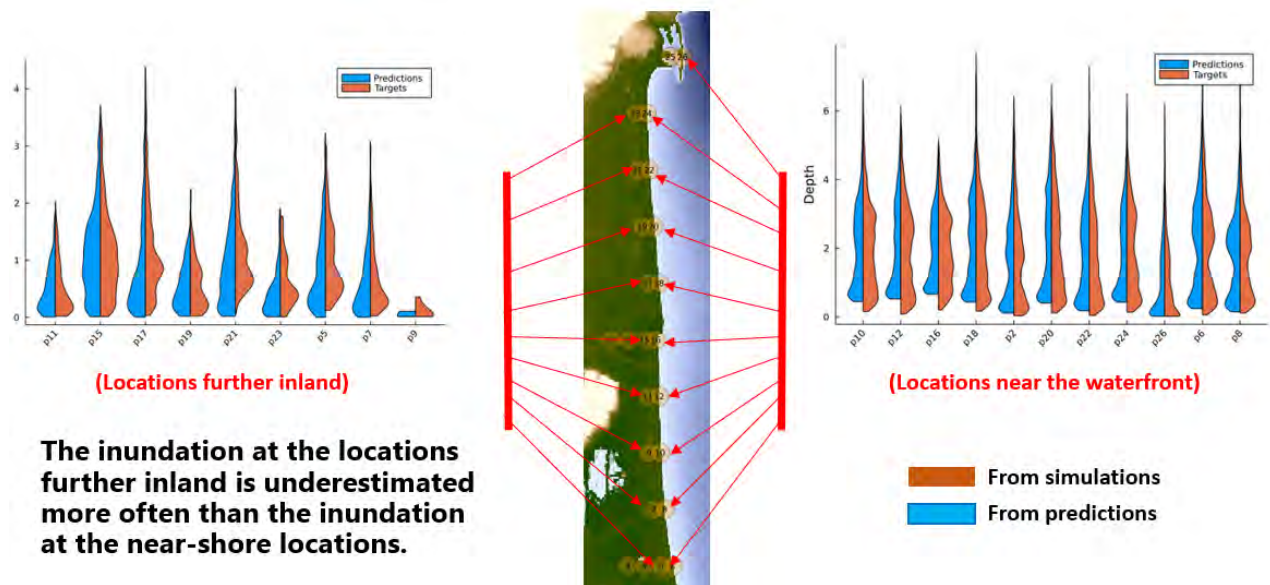


Figure 25. Comparison of the maximum inundation heights at selected locations in the Bay of Catania between the physics simulations (red) and the encoder-decoder predictions (blue). The violin plots show the distributions over the test set of maximum flow-depth (i.e. the maximum inundation height with the elevation subtracted) for a preliminary random training set. A completely accurate emulator would result in identical distributions on each side of the violin plots. The performance for many of the locations closest to the waterfront is very promising with fairly symmetrical distributions. For many of the more inland locations, the physics simulations predict higher inundations than the ML emulator. The reasons may be complex, but the simplest explanation is that there is inundation at the more inland locations at far fewer scenarios in the training set.

We emphasize that ML-prediction for tsunami is necessarily site-specific. We cannot train a model for one stretch of coastline and use it to predict the inundation at another location. The performance of the models will depend both upon the architecture and parameters of the models and on the available training sets. For the emulator to have a useful purpose, we require of course accuracy in the predictions (overestimation and underestimation of the inundation are equally undesirable) and that there is a significant reduction of the computation cost/time-to-solution. If the training set needed were to be too large a proportion of the total number of scenarios needed, this would defeat the point of employing an emulator.

#### 4.2.2. Tsunami inundation with EDDL

Several codes have been developed using the EDDL library (C++ version) and MPI to predict results of tsunami simulations for the classification and regression problems that were described in the item “ML/AI: Tsunami Modelling Emulation” in section 3.2 of the deliverable D6.3.

#### 4.2.3. Alert Levels

The classification problem consists of predicting the alert level in a certain area given the nine Okada parameters (longitude, latitude, depth, fault length, fault width, strike, dip, rake and slip) of the earthquake that causes the tsunami.

A code has been implemented in EDDL and MPI to train a multilayer perceptron (MLP) network given the inputs and outputs of the network. The input consists of the nine Okada parameters of all the samples, and the output is the alert level obtained for all the samples. Different number of

alert levels with different discretizations are supported. It is possible to specify the desired partition for the train, validation and test sets, the architecture of the network, the batch size, the patience for the early stopping, the initial learning rate, and, optionally, the final learning rate if the validation loss gets stalled during the training process. The categorical cross entropy or the accuracy can be used as the loss function. It is possible to run multiple trainings in parallel, each one on a different GPU, where the network weights are randomly initialized for each training. At the end of the process, the best obtained model (based on the value of the validation loss) is saved in ONNX format.

A code to infer results given a model in ONNX format and inputs of the network has also been implemented using EDDL. The result is a text file containing the alert level predicted for all the input samples using the provided model.

Finally, a python script has been developed to obtain the input and output parameters of the network given a set of NetCDF result files obtained with Tsunami-HySEA, the desired discretization of the alert levels, and the point in which to predict the alert level. It is possible to compute the alert level from the free water surface (for offshore points) or from the water thickness (for onshore points). The obtained files containing the network inputs and outputs are the ones that will be used in the training process.

#### **4.2.4. Maximum Water Height**

The regression problem consists of predicting the maximum water height reached at several points given the nine Okada parameters of the earthquake that causes the tsunami.

We have developed the same codes described in section 4.3.1 applied to this regression problem. The training code is implemented in EDDL and MPI to train a MLP network given the inputs and outputs of the network. The network inputs are again the nine Okada parameters, and the output is the maximum water height achieved in all the considered points. Any number of points can be used. It is possible to specify the values of the same hyperparameters as in the classification problem. The mean squared error is used as the loss function. Multiple trainings can be performed in parallel using different GPUs, and the best obtained model is saved in ONNX format.

The inference code receives a model in ONNX format and network inputs, and it outputs a file with the maximum water height predicted in all the points for which the model has been trained.

Finally, a python script has been implemented to obtain the input and output parameters of the network given a set of NetCDF result files obtained with Tsunami-HySEA, and the location of all the desired points (longitude and latitude) in which to predict the maximum water height. As output parameters of the network it is possible to obtain the maximum free water surface (for offshore points) or the maximum water thickness (for onshore points).

#### **4.2.5. Tsunami emulator based on decision-trees**

We present here a machine learning approach based on regression trees to model and forecast tsunami simulations. The input data of the analysis is a set of simulations forming an ensemble that describes potential regional impact of tsunami source scenarios in a given source area, where each instance is described by input parameters describing the geometry/kinematic of faults triggering the earthquake (tectonic region, magnitude, longitude, latitude, etc.), and output values (simulation results) corresponding to the estimated heights of tsunami waves at several target

points in front of the coast close. Given such simulation data, the approach aims at training a predictive model that can be applied to any potential tsunami source in the area to forecast the height of waves at all the target locations. This model, being based on a regression tree, is computationally light and fully-readable by domain experts, allowing both to inspect the underlying rules for checking its internal consistency and learn from the selected features the leading source characteristics in the area.

Before describing the approach, we report a definition of the main concepts underlying the proposed solution and the objectives of the analysis.

**Input Data.** Let  $D$  be a dataset collecting simulation data instances,  $D = \{d_1, d_2, \dots, d_N\}$ , where each  $d_i$  is a tuple representing a tsunami simulation. Specifically, each data tuple is modeled by a  $(I + H)$ -dimensional attribute vector  $A = \{a_1, \dots, a_I, a_{I+1}, \dots, a_{I+H}\}$ , where the first  $I$  attributes describe the input parameters and the remaining  $H$  attributes refer to the output results of the simulation. In the specific case related to tsunami simulation, the dataset of precomputed scenarios is derived from [regtree1, regtree2], in which individual scenarios are associated to the attributes  $a_1, \dots, a_I$ , as detailed in the following:

- *region*: the seismotectonic regions as defined within the NEAMTHM18 project; the regionalization was built following basic plate tectonics principles and by refining or adapting the regionalization of the European seismic hazard model ([regtree2]).
- *magnitude*: the moment magnitude of the earthquake.
- *longitude*: longitude of the place the earthquake occurs.
- *latitude*: latitude of the place the earthquake occurs.
- *depth of the top*: depth of the upper edge of the fault plane.
- *strike*: angle indicating the orientation in space of the fault defined as the clockwise angle (turning around the normal outgoing from the earth's surface) between the North direction and the positive strike direction.
- *dip*: the dip angle is the angle less than or equal to  $90^\circ$  between the horizontal plane and the fault plane: it gives the direction of the movement on the fault.
- *rake*: the angle that the direction of relative movement (of the hanging wall with respect to the foot-wall of the fault rupture) forms with the direction of strike, measured counterclockwise from the strike direction.
- *area*: the surface area of the rupture fault plane ( $KM^2$ ).
- *length (KM)*: the dimension of the fault plane in the along-strike direction
- *average slip*: the average amount of relative movement between the two fault surfaces.

These attributes describe the geometry and kinematic of faults generating the earthquake (simulation inputs). In the precomputed scenarios, three attributes (area, length and average slip) have been deterministically defined from magnitude adopting appropriate scaling laws [regtree2]. Since they do not contain any additional information, only the magnitude attribute is kept, while area, length and average slip are not part of the input parameters set.



Further attributes correspond to the expected maximum heights of tsunami waves at the target points in front of the coast (simulation outputs). In particular, the attributes  $\{a_{I+1}, \dots, a_{I+H}\}$  correspond to the output fields  $hmax_1, \dots, hmax_H$  of the simulations, where each  $hmax_h$  is the maximum height of tsunami waves estimated by the simulator at the  $h^{\text{th}}$  target point ( $h = 1, \dots, H$ ).

**Output Models.** Our goal is to find a regression model for reliably predicting the results of a new simulation (i.e.,  $hmax_1, \dots, hmax_H$  values), given an input parameter value setting (i.e., region, magnitude, ..., average slip). Formally, we want to extract a set  $F_{hmax}$  of tsunami predictors,  $F_{hmax} = \{F^1_{hmax}, \dots, F^H_{hmax}\}$ , where each function  $F^h_{hmax}: R^I \rightarrow R$ , given a specific input parameter setting, forecasts the maximum height of tsunami waves at the  $h^{\text{th}}$ -target location.

**The approach.** The dataset to be analyzed is the set  $D$  of collected Tsunami simulation data (represented in the previous described format). The approach computes and returns the set  $F_{hmax}$  of Tsunami predictors. The workflow is composed of three main steps (see Figure 26), as described in the following.

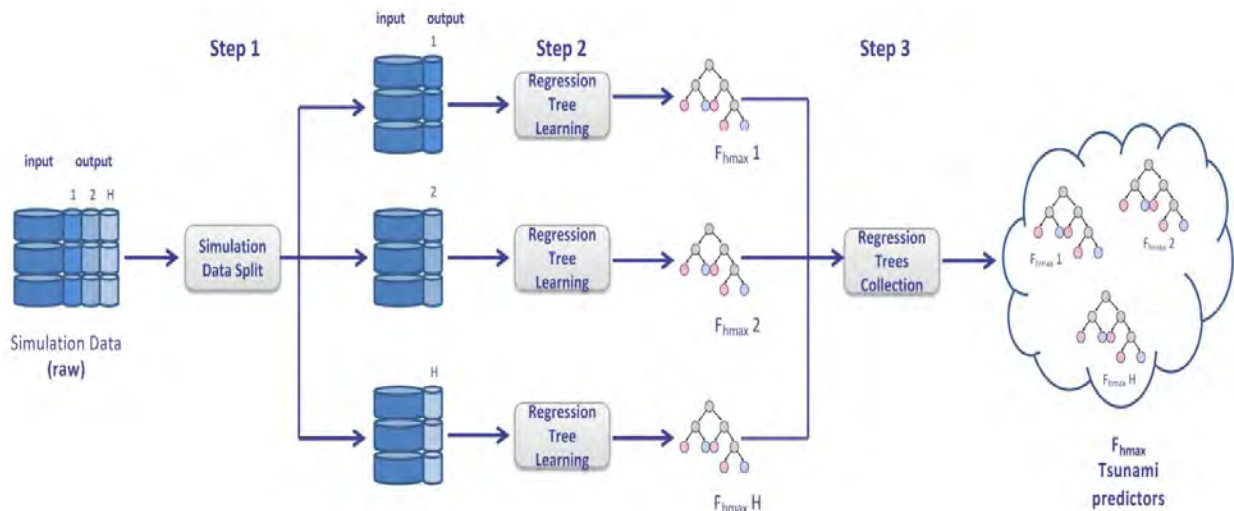


Figure 26. Regression tree learning workflow

The first step is aimed at splitting the original simulation data in a vertical way, with respect to each specific target location output. In other words, simulation inputs and the  $h^{\text{th}}$ -output data are gathered in the  $h^{\text{th}}$ -dataset, for  $h = 1, \dots, H$ . At the end of this step,  $H$  different datasets are produced, each one containing a vertical projection of  $D$  on the  $h^{\text{th}}$ -target output. The second step consists in the induction of the predictive models, for each target location. In the workflow, this is done by running  $H$  regression tree algorithm instances, each one taking in input a dataset built at the previous step. The result consists of  $H$  predictive models  $F^1_{hmax}, \dots, F^H_{hmax}$  whereas the  $h^{\text{th}}$ -model represents the tsunami predictor for the  $h^{\text{th}}$ -target location. Finally, the third step is aimed at collecting the predictive models extracted at step two. The final result is the whole set  $F_{hmax}$  of tsunami predictors, which can be used to forecast tsunami waves at run-time.



### 4.3.PTF based on High resolution HySEA simulations

#### 1) HySEA for high-resolution simulations of the tsunami propagation and inland inundation

One crucial enhancement being incorporated into the workflow is the implementation of two-way nested meshing for high-resolution simulations. With this incorporation, the workflow takes a significant leap forward in its capabilities. By adopting this capability, the workflow not only facilitates standard propagation simulations but also opens the door to conducting high-resolution simulations and comprehensive studies on coastal inundation. The two-way nested meshing empowers the model to refine spatial resolution in specific regions of interest, leading to more accurate and precise results. This enhancement provides invaluable insights into the complex behavior of tsunamis during coastal inundation events. Consequently, the workflow becomes more versatile and expansive, enabling researchers and practitioners to explore a wide array of scenarios and make well-informed decisions regarding tsunami risk assessment and effective mitigation strategies. The incorporation of high-resolution simulations elevates the workflow's capacity to unravel intricate coastal processes, thereby bolstering its relevance and significance in tsunami research and preparedness efforts.

As a first test, a high-resolution study area has been established in the Majorca Port, located in the Balearic Islands, Spain. To achieve this, the construction of four nested mesh levels was carried out, ranging from the coarser ambient mesh at 640 meters to the finest mesh at an impressive 5-meter resolution. Additionally, the parameter files were adapted to facilitate the simulations launched from the Tsunami-HySEA software.

The implementation of the nested mesh approach allows us to finely refine the spatial resolution in the study area, providing a detailed representation of the coastal environment. By focusing on the Mallorca Port, we can gain in-depth insights into the specific behavior of tsunamis in this critical location, understanding the intricacies of tsunami propagation and inland inundation at a level of detail previously unexplored.

Indeed, with this type of high-resolution test, the goal is to validate the results by comparing them with historical records of past tsunami events. This validation process aims to demonstrate the efficacy and accuracy of the numerical models employed within the workflow.

#### 2) Adaptation of the PTF PyCOMPSS workflow to the HySEA high-resolution simulation

The workflow built initially for the evaluation of the forecast at predefined locations on a large mediterranean grid, as described in Section 3.2.1, has been now adapted and tested for the site of Majorca.

The task1 of the ensemble manager was kept the same, the task 2-3 for the configuration of HySEA parameters and tasks 4-7 to produce PTF aggregation were modified to match the High-Resolution simulations. The main changes were done for the preparation of the input files of HySEA and for the postprocessing of the results. Indeed the objective was not to compute Hazard Curves at predefined locations on the maps (with, for example, 1,000 points as it is in the classic PTF WF) but for each point of the finest grid (La Palma with 5m resolution).

To do so the Maximum Wave Amplitude grid outputs (200 grids corresponding to 200 simulations) were read and flattened by a python program in order to merge all results with their corresponding probabilities and to make the necessary forecast operations, see Figure 27. The final matrix corresponds to the hazard curve values (23 values) for each point of the grid (13,081,600 points).

From this matrix, forecast mean and percentiles can be extracted, and can be used to reconstruct the forecast maps as NetCDF output.

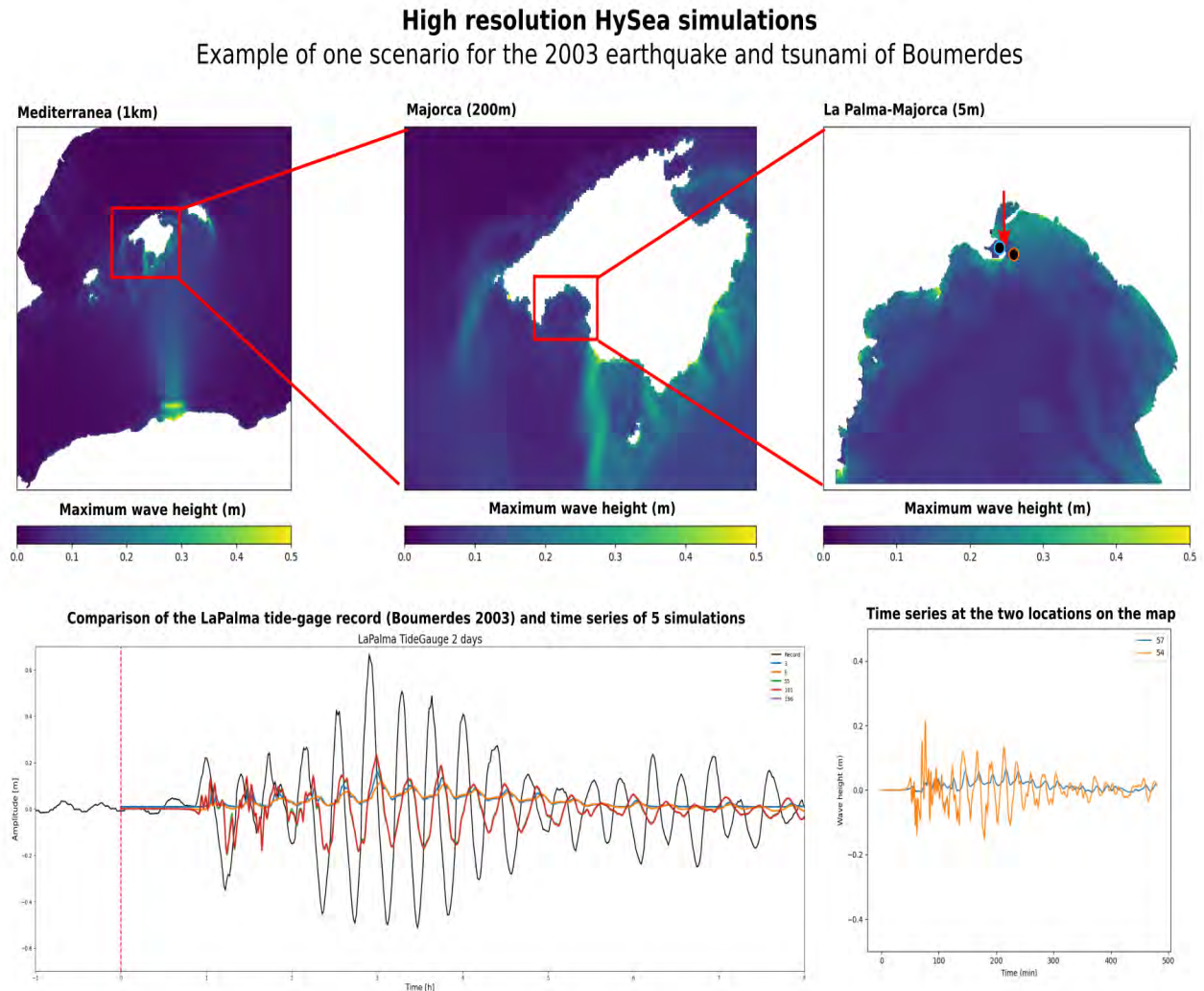


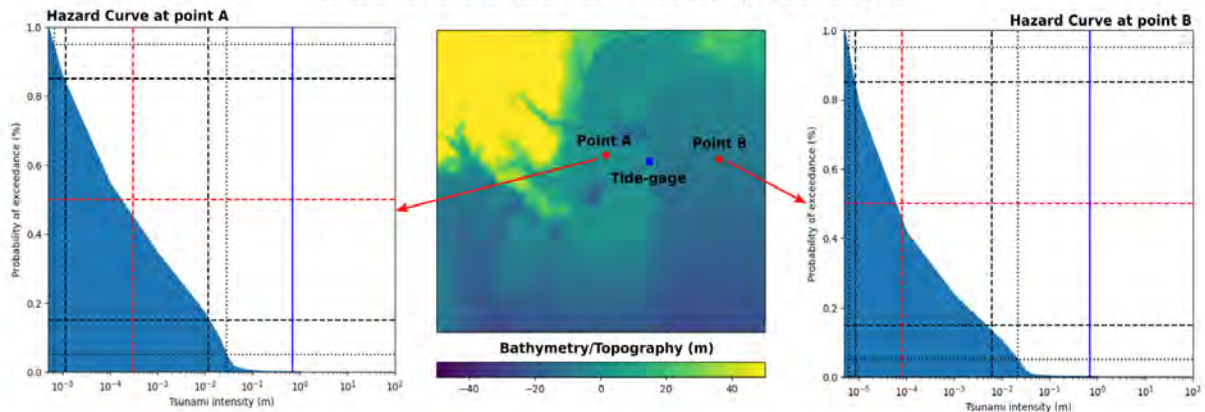
Figure 27. Example of a 8 hours high resolution simulation of the 2003 Boumerdes earthquake on the LaPalma grid set with synthetic time-series at different points of the map.

The size of the matrix being particularly large for the necessary mathematical operation (calculation of lognormal distributions, array products,...). The *-highmemory* option of the BSC job system was used so that this high memory usage could be allowed and the calculations be completed.

This HR\_PTF\_WF was tested for 200 scenarios for the 2003 Boumerdes event on the LaPalma grid. It took around 72h to complete the calculation with 10 nodes and generated an output folder of around 500GO, see Figure 28.

### Aggregation of the results of 500 simulations for the forecast calculation

Calculation of the hazard curves at each point of the grid



Creation of mean or percentiles maps (mean, p5, p95) - Extraction of the values at specific locations

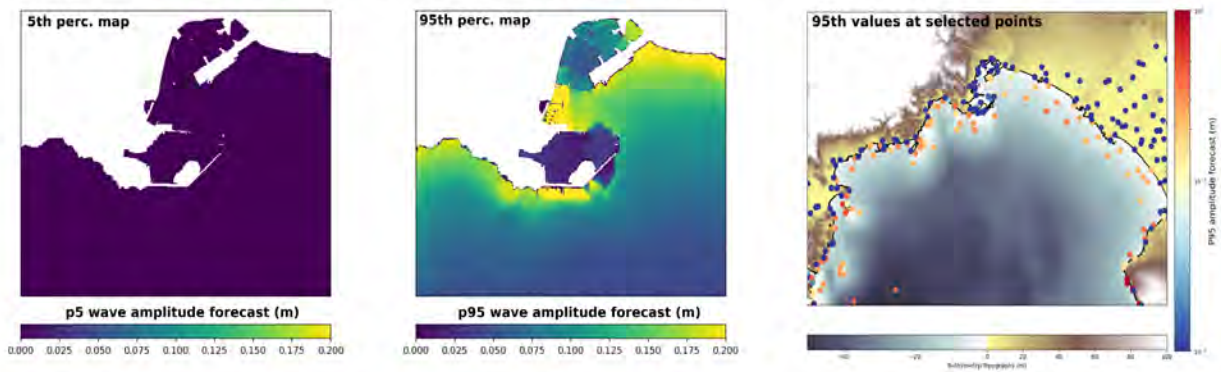


Figure 28. 4.8: PTF results of the 2003 Boumerdes earthquake, with a workflow run for 200 scenarios. The different percentiles can be mapped and the hazard curves can be extracted from each point of the LaPalma grid.

## 5. Workflow access codes

The following table shows the links in the eFlows4HPC official GitHub repository to access the codes to the workflows described in this deliverable.

Table 3. : Links to access to the different workflows described in this deliverable.

Workflow	Link
PTF/FTRT	<a href="https://github.com/eflows4hpc/ptf_workflow">https://github.com/eflows4hpc/ptf_workflow</a>
UCIS4EQ	<a href="https://github.com/eflows4hpc/ucis4eq">https://github.com/eflows4hpc/ucis4eq</a>
MLESmap	<a href="https://github.com/eflows4hpc/mlesmap">https://github.com/eflows4hpc/mlesmap</a>
Emulators for Tsunami propagation	<a href="https://github.com/eflows4hpc/ML_tsunamis">https://github.com/eflows4hpc/ML_tsunamis</a>

## 6. Conclusions

In this deliverable we have presented the second iteration of the workflows that comprise Pillar III, namely UCIS4EQ and PTF. The work presented here summarizes the efforts across the three phases and two iterations of development, in collaboration with WPs 1 to 3. The existing workflows are now faster, more robust, and more capable, thanks to incorporating components of the eFlows4HPC software stack, namely PyCOMPSs, Ophidia, dislib or EDDL. They are now available by means of the automated deployment of the HPCWaaS. On top of that, components based on machine-learning applications, not foreseen at the time of writing the proposal, have been developed during the project's lifetime and are now candidates for integration into their parent workflows.

## 7. List of figures and tables

Table 1. Status of the execution requirements in the UCIS4EQ Workflow of Pillar III (D6.1) and the current status at the Iteration 2 - Phase 3 (M19-M30).....	12
Table 2. Computer resources per each station in a Cybershake execution for the Iceland use case.....	30
Table 3. : Links to access to the different workflows described in this deliverable. ....	40
Figure 1. A general overview of the UCIS4EQ workflow .....	5
Figure 2. Schematic differences between the ensemble methods used in UCIS4EQ, the statistical CMT (Monterrubio et al., 2021) and SeisEnsMan.....	6
Figure 3. Representation of the SeisEnsMan module and its interaction with the UCIS4EQ building blocks .....	7
Figure 4. Required information to set the new Mexican earthquake towards Task 6.5 .....	8
Figure 5. Task definition with PyCOMPSs for an asynchronous microservice invocation.....	9
Figure 6. Code Snippet of the HPC workflow with PyCOMPSs.....	10
Figure 7. Graph of TOSCA pipelines used by UCIS4EQ.....	11
Figure 8. Orchestration of the PTF workflow using PyCOMPSs .....	15
Figure 9. PTF workflow PyCOMPSs script: PyCOMPSs tasks and functions' definition.....	16
Figure 10: PTF workflow PyCOMPSs script: workflow main script .....	17
Figure 11. PyCOMPSs graph of a full PTF workflow run (for a small example of 16 scenarios) with all the tasks and dependencies.....	18
Figure 12. Convergence of the PTF results for multiple Monte-Carlo sampled ensembles .....	19
Figure 13. Files produced progressively by the convergence module at each delivery step .....	20
Figure 14. Example of PTF results (Hazard Curve and extracted percentiles and mean values) obtained by the classic calculations (black line), by the update using the Focal Mechanism (blue line) and by the update using the Tsunami Data (orange line) for the 2023 Turkey earthquake .....	22
Figure 15. Application menu and topology editing functionality with Alien4Cloud. ....	23
Figure 16. Excerpt from the TOSCA description of the initial PTF Workflow. ....	23
Figure 17. Menu of federated HPC facilities on which the workflow could be deployed. ....	24
Figure 18. Specification of submission parameters and arguments for deployment of workflow on the selected HPC resource within Alien4Cloud. ....	25
Figure 19. Schematic diagram of the PTF workflow with the indication on where and how Ophidia can improve the post-processing tasks. ....	26
Figure 20. Internal steps for the computation of the variables for each scenario. ....	27
Figure 21. MLESmap methodology for the prototype developed at Iteration 1 - Phase 2 (M6-M18).....	28



Figure 22. MLESmap workflow developed at iteration 3 phase 2 (M19-M30). Pink boxes show the steps that must be executed in a HPC environment.....	29
Figure 23. Sketch of an encoder-decoder architecture for predicting high-resolution maps of maximum onshore tsunami inundation from the (far cheaper computationally) offshore time-series. ....	32
Figure 24. The percentage of earthquake tsunami scenarios in a PTHA study [S3] exceeding the flow depth indicated across the highest resolution inundation grid for the Bay of Catania, Sicily.....	33
Figure 25. Comparison of the maximum inundation heights at selected locations in the Bay of Catania between the physics simulations (red) and the encoder-decoder predictions (blue). ....	34
Figure 26. Regression tree learning workflow .....	37
Figure 27. Example of a 8 hours high resolution simulation of the 2003 Boumerdes earthquake on the LaPalma grid set with synthetic time-series at different points of the map. ....	39
Figure 28. 4.8: PTF results of the 2003 Boumerdes earthquake, with a workflow run for 200 scenarios.....	40

## 8. Acronyms and abbreviations

<b><i>Term or abbreviation</i></b>	<b><i>Description</i></b>
<b>AI</b>	Artificial Intelligence
<b>ChEESE</b>	Center of Excellence for Exascale in Solid Earth
<b>D1.1</b>	Deliverable 1.1 Requirements, metrics and architecture design
<b>DA</b>	Data Analytics
<b>DAG</b>	Directed Acyclic Graph
<b>DAL</b>	Data Access Layer
<b>EQ</b>	Earthquake
<b>FAIR</b>	Findable, Accessible, Interoperable, Reusable
<b>FTRT</b>	Faster-Than-Real-Time (FTRT)
<b>GMPE</b>	Ground Motion Prediction Equations
<b>HPC</b>	High Performance Computing
<b>HPDA</b>	High Performance Data Analytics
<b>HPCWaaS</b>	HPC Workflow as a service
<b>HySEA</b>	Hyperbolic Systems and Efficient Algorithms
<b>ML</b>	Machine Learning

<b>MLESmap</b>	Machine-Learning based Estimator for ground motion Shaking maps
<b>NN</b>	Neural Network
<b>PTF</b>	Probabilistic Tsunami Forecast
<b>QoS</b>	Quality of Service
<b>TWCs</b>	Tsunami Warning Centers
<b>UC</b>	Urgent Computing
<b>UCIS4EQ</b>	Urgent Computing Integrated Services for EarthQuakes
<b>WM</b>	Workflow Manager
<b>WP1</b>	Work Package 1

## 9. References

- [S1] J. Selva et al., ‘Probabilistic tsunami forecasting for early warning’, Nat. Commun., vol. 12, no. 1, 2021, doi: 10.1038/s41467-021-25815-w
- [S2] A. Folch et al., ‘The EU Center of Excellence for Exascale in Solid Earth (ChEESE): Implementation, results, and roadmap for the second phase’, Future Gener. Comput. Syst., p. S0167739X23001401, Apr. 2023, doi: 10.1016/j.future.2023.04.006.
- [S3] S. J. Gibbons et al., ‘Probabilistic Tsunami Hazard Analysis: High Performance Computing for Massive Scale Inundation Simulations’, Front. Earth Sci., vol. 8, no. December, pp. 1–20, Dec. 2020, doi: 10.3389/feart.2020.591549.
- [S4] Taroni M, Selva J (2021), A Testable Worldwide Earthquake Faulting Mechanism Model Seismol. Res. Lett., 92 (6): 3577-3585, <https://doi.org/10.1785/0220200445>
- [S5] Selva J, Tonini R, Molinari I, Tiberti MM, Romano F, Grezio A, Melini D, Piatanesi A, Basili R, Lorito S (2016), Quantification of source uncertainties in Seismic Probabilistic Tsunami Hazard Analysis (SPTHA), Geophys. J. Int. 205, 1780–1803, doi:10.1093/gji/ggw107
- [S6] 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, Gürbüz M, Pekcan O, Yalçiner A, Canals M, Lastras G, Agalos A, Papadopoulos G, Triantafyllou I, Bencheckroun 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, González Vida JM, González M, Macías Sánchez J, Meletti C, Ozer Sozdinler C, Pagani M, Parsons T, Polet J, Power W, Sørensen W, Zaytsev A (2021), The making of the NEAM Tsunami Hazard Model 2018 (NEAMTHM18), Frontiers in Earth Science, 9, 82 <https://doi.org/10.3389/feart.2020.616594>

- [S7] Monterrubio-Velasco, J. E. Rodriguez, M., Pienkowska, M., L. Mingari, and de la Puente, J.: UCIS4EQ applied to Mediterranean earthquakes, MedGU 2022, Marrakech, Morocco, 27–30 Nov 2023, MedGU abstracts book 2022 - 95, Springer.
- [S8] Monterrubio-Velasco, M., Pienkowska, M., and de la Puente, J.: UCIS4EQ applied to the South Iceland region, EGU General Assembly 2023, Vienna, Austria, 24–28 Apr 2023, EGU23-12982, <https://doi.org/10.5194/egusphere-egu23-12982>, 2023.
- [S9] Monterrubio-Velasco, et al., (2021). Source Parameter Sensitivity of Earthquake Simulations assisted by Machine Learning, EGU General Assembly 2021, online, 19–30 Apr 2021, EGU21-5995, <https://doi.org/10.5194/egusphere-egu21-5995>, 2021.
- [S10] Rojas O., Rodriguez J.E., Monterrubio M., Callaghan S., de la Puente J., et al. Insights on Physics-based Probabilistic Seismic Hazard Analysis in South Iceland using CyberShake. AGU fall meeting 13-17 December 2021.
- [S11] Monterrubio-Velasco, M., Modesto, D., Callaghan, S., and de la Puente, J.: Machine Learning based Estimator for ground Shaking maps, Galileo Conference: Solid Earth and Geohazards in the Exascale Era, Barcelona, Spain, 23–26 May 2023, GC11-solidearth-50, <https://doi.org/10.5194/egusphere-gc11-solidearth-50>, 2023b.
- [S12] Monterrubio-Velasco M., Callaghan S., Modesto D., de la Puente J.: Machine Learning based Estimator for ground Shaking maps applied to the Los Angeles basin region. IUGG23-0719. International Union of Geophysics and Geodesy General Assembly 2023.
- [S13] Stallone, A., Cordrie L., Michelini A., Selva J.: A Python algorithm for ensemble initialization and reduction in urgent computing applications and probabilistic shakemaps retrieval. S09p-363. International Union of Geophysics and Geodesy General Assembly 2023.
- [regtree1] Selva J, Lorito S, Volpe M, Romano F, Tonini R, Perfetti P, Bernardi F, Taroni M, Scala A, Babeyko A, Løvholt F, Gibbons SJ, Macías J, Castro MJ, González-Vida JM, Sánchez-Linares C, Bayraktar HB, Basili R, Maesano FE, Tiberti MM, Mele F, Piatanesi A, Amato A., Probabilistic tsunami forecasting for early warning, *Nature Communications* 12 (1) (2021) 56–77.
- [regtree2] 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, Gürbüz M, Pekcan O, Yalçiner A, Canals M, Lastras G, Agalos A, Papadopoulos G, Triantafyllou I, Benchekrone S, Agrebi Jaouadi H, 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, González M, Macías Sánchez J, Meletti C, Ozer Sozdinler C, Pagani M, Parsons T, Polet J, Power W, Sørensen M and Zaytsev A, The making of the neam tsunami hazard model 2018 (neamthm18), *Frontiers of Earth Science* 8 (1) (2021) 56–77.
- [hysea1] Castro, M.J.; Ferreiro, A.; García, J.A.; González, J.M.; Macías, J.; Parés, C.; Vázquez, M.E. On the numerical treatment of wet/dry fronts in shallow flows: Applications to one-layer and two-layer systems. *Math. Comp. Model* 2005, 42, 419–439.
- [hysea2]. Castro, M.J.; González, J.M.; Parés, C. Numerical treatment of wet/dry fronts in shallow flows with a modified Roe scheme. *Math.Mod. Meth. App. Sci.* 2006, 16, 897–931.

- [hysea3]. Castro, M.J.; Chacón, T.; Fernández-Nieto, E.D.; González-Vida, J.M.; Parés, C. Well-balanced finite volume schemes for 2D non-homogeneous hyperbolic systems. Applications to the dam break of Aznalcóllar. *Comp. Meth. Appl. Mech. Eng.* 2008, 197, 3932–3950.
- [hysea4]. Asunción, M.; Castro, M.J.; Fernández-Nieto, E.D.; Mantas, J.M.; Ortega, S.; González-Vida, J. M. Efficient GPU implementation of two waves TVD-WAF method for the two-dimensional one layer shallow water system on structured meshes. *J. Comput. Fluids* 2013, 80, 441–452
- [hysea5]. Gallardo, J.M.; Parés, C.; Castro, M. On a well-balanced high-order finite volume scheme for shallow water equations with topography and dry areas. *J. Comp. Phys.* 2007, 227, 574–601.
- [hysea6] Macías, J.; Castro, M.J.; Escalante, C. Performance assessment of Tsunami-HySEA model for NTHMP tsunami currents bench-marking. Laboratory data. *Coast. Eng.* 2020, 158, 103667.
- [hysea7] Macías, J.; Castro, M.J.; Ortega, S.; González-Vida, J.M. Performance assessment of Tsunami-HySEA model for NTHMP tsunami currents benchmarking. Field cases. *Ocean Model* 2020, 152, 101645.
- [hysea8] 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 Appl. Geophys.* 2017, 174, 3147–3183.
- [ens1] Cordrie, Louise, et al. Using available and incoming data for reducing and updating seismic source ensembles for probabilistic tsunami forecasting (PTF) in early-warning and urgent computing. No. EGU23-12363. Copernicus Meetings, 2023.
- [ens2] Cordrie, L., Selva, J., Bernardi, F., Tonini, R., Macías Sánchez, J., Sánchez Linares, C., Gibbons, S., and Løvholt, F.: Complete workflow for tsunami simulation and hazard calculation in urgent computing using HPC services, Galileo Conference: Solid Earth and Geohazards in the Exascale Era, Barcelona, Spain, 23–26 May 2023, GC11-solidearth-14, <https://doi.org/10.5194/egusphere-gc11-solidearth-14>, 2023