D5.4 Pillar II - Iteration 2 Software Release

Version 1.0

Documentation Information

Contract Number	9555558
Project Website	www.eFlows4HPC.eu
Contractual Deadline	31.08.2021
Dissemination Level	[PU]
Nature	OTHER
Author	Alessandro D’Anca (CMCC)
Contributors	Gabriele Accarino (CMCC), Sonia Scardigno (CMCC), Donatello Elia (CMCC), Davide Donno (CMCC), Rohan Ahmed (BSC), Bruno de Paula Kinoshita (BSC), Suvarchal K. Cheedela (AWI), Nikolay Koldunov (AWI)
Reviewer	Scott Keating (ETHZ)
Keywords	Earth System Models, workflows, data analysis, machine learning

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	Initial ToC
V0.2	First version ready for internal review
V0.3	Second version reviewed by internal reviewers
V0.4	Third version revised after internal review
V1.0	Final version ready for submission

Table of Contents

1. Executive Summary

This deliverable describes the status of the development activities performed in the context of Pillar II of the eFlows4HPC project at the end of Iteration 2 – Phase 3. Two workflows have been taken into account, both belonging to the climate science field and involving Earth System Models results analysis:

- The dynamic ESM workflow, with the aim to integrate dynamic access to model results and online-based approaches to prune ensemble simulation members during the model execution;
- The feature extraction workflow, with the aim to integrate HPDA and ML approaches on ESM results to analyze extreme events such as Heat and Cold Waves and Tropical Cyclones.

The document provides an overview of the main achievements reached during this phase, followed by a detailed description of the status and the last advancements concerning the two aforementioned workflows.

2. Introduction

The eFlows4HPC Iteration 2 – Phase 3 concerns the second version of the Pillars' workflows. At this stage, in the context of Pillar II, almost all the different modules constituting the two workflows have been developed and deployed. Both workflows exploit the eFlows4HPC tools and software (i.e., Ophidia[1], Hecuba[2], PyCOMPSs[3], etc.), and both have been integrated with the expected software stack for a first prototypal end-to-end execution.

In addition, further advancements have been made with respect to the optimization of existing functionalities and the development of new ones to better support the objectives of the project and the needs of the climate community[4].

3. Summary of achievements

In this section the main achievements with respect to Iteration 1 – Phase 2 are highlighted; they regard both workflows considered in Pillar II and concern firstly the novel capabilities developed to support the Pillar II use cases, then the optimization of functionalities implemented, and finally the integration with the eFlows4HPC HPCWaaS software stack by means of the related software and frameworks (e.g., Alien4Cloud[5] PyCOMPSs, Yorc[6], etc.).

3.1. Dynamic ESM workflow

Running a coupled Earth System Model (ESM) on a supercomputer can be a challenging task as ESM's consist of several steps to prepare, run, and refine and obtain the output data (post-processing and diagnostics stages [7, 8]).

With the complexity of ESMs in view, the work was decomposed into incremental parts, running and building workflows with ocean only, FESOM2 model and then integrating coupled OpenIFS-FESOM2 into the developed workflows. We can summarize the achievements since the last deliverable (D5.3) as follows:

- Development and deployment of the workflow using FESOM2 ocean model on MareNostrum 4 using Alien4Cloud (more on this in Section 4).
- Development of an extended version of HECUBA to simplify specification of data models used to ingest climate model data into Cassandra[9].
- Integration of PyCOMPSs to support pruning ensemble.
- Deployment of the OpenIFS-FESOM2 (AWICM[10]) coupled model in MareNostrum 4, and its initial integration into the workflow as a component.

These are described in detail in Section 4, along with plans for the final phase of the project.

3.2. Statistical analysis and feature extraction workflow

The statistical analysis and feature extraction workflow of Pillar II has the aim to integrate ML and HPDA features in a single end-to-end workflow in order to analyze the occurrence of extreme events (specifically Heat Waves and Tropical Cyclones) starting from the output of the CMCC-CM3 climate model.

With respect to Iteration 1, the following advancements and optimizations have been achieved. In Section 5 they will be analyzed more in details:

- We applied the STREAM IN parameter to the definition of the initial PyCOMPSs task allowing the use of the PyCOMPSs streaming interface to better manage the entire workflow execution.
- The IBTrACS (recent and historical tropical cyclone datasets) filtering has been improved, selecting 6-hourly records.
- 13 models have been trained and fine-tuned for the ML based TC detection task on historical data, as well as their classification and localization metrics on the test set.
- The TSTORMS[11] tool for the deterministic TC detection has been integrated in the overall PyCOMPSs workflow.
- Exploiting Alien4Cloud and YORC the workflow has been integrated in the eFlows4HPC HPCWaaS framework.

4. Dynamic ESM simulation workflow

This section provides an overview of the work achieved in the Dynamic ESM simulation workflow development by the end of the second iteration - Phase 3. The current execution flow of the ESM dynamic workflow is illustrated in figure 1.

During the first iteration, the FESOM2 ocean model, Hecuba, and PyCOMPSs were integrated into the workflow, and this was deployed on MareNostrum 4. The workflow in this iteration was executed as a conventional SLURM[12] batch script.

In the second development iteration of WP5, the coupled model was integrated with the workflows using ocean only configurations. This phase also included containerizing workflow components for eFlows4HPC software stack, used to create the HPCWaaS application. In the following subsections we describe different blocks of this phase and list next steps until completion of the project.

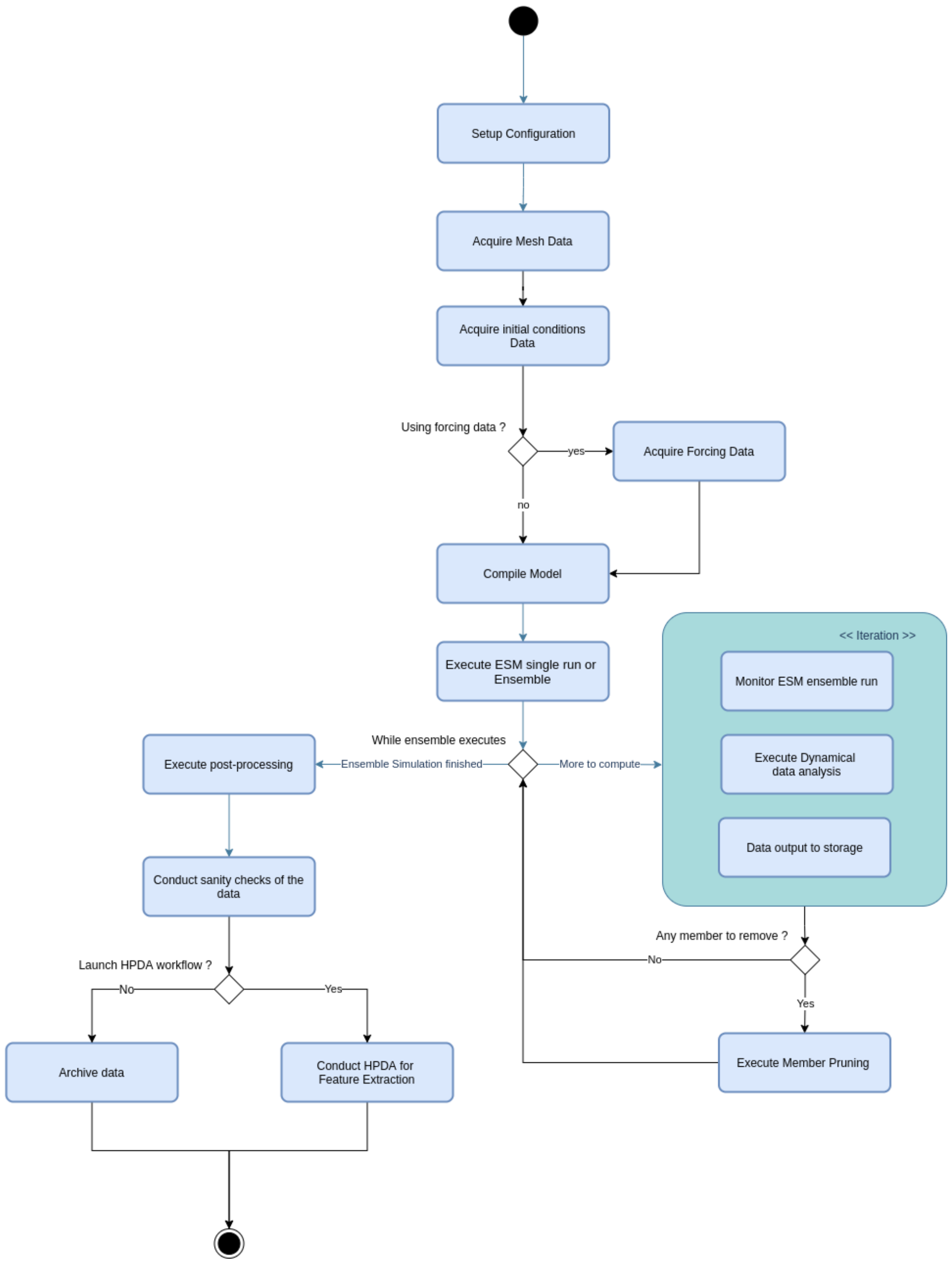


Figure 1. Activity diagram displaying the overall ESM dynamic workflow.

4.1. ESM initialization block

One of the prerequisites for running the ESM workflow, as mentioned in deliverables D5.1, D5.2, and D5.3, is to prepare several datasets required for the model run. These datasets are included in the general ensemble settings defined for the run, which are centralized in the master ensemble configuration file (figure 2, changes from iteration 1 in bold).

In D5.4 the master configuration file received new settings to define the number of cores needed for each model component, and other variables that manage the coupling and OpenIFS settings. This is depicted in figure 2, where several "export" statements are highlighted.

```
        module load COMPSs/eflows4hpc
        module use /apps/HECUBA/modulefiles
        module load Hecuba/2.0 #
        module load impi/2018.4
        module load netcdf/4.4.1.1
        module load eccodes/2.8.0
        module load hdf5/1.8.19

        #module load Hecuba/1.0_api
        # Hecuba already loads a special version of Python so not needed at this stage
        #module load python
        # experiment configuration
        # only parameter - #CORES
        #export FESOM_CORES=$1
        # possible values: bsc_es/debug
        export QOS=debug
        export MEMBERS=3
        export USE_HECUBA="TRUE"
        # 7 nodes per ensemble  member
        export NODE_ALLOCATION=7
        [[ "$1" == "TRUE" ]] && USE_HECUBA="TRUE" || USE_HECUBA="FALSE"
        echo "Number of cores: ${FESOM_CORES}"
        echo "Number of nodes to be used: ${NODE_ALLOCATION}"

        #added by support to prevent the segmentation fault
        ulimit -Ss unlimited

        ##########  components and core configuration ##########

        export OIFS="./master.exe"
        export OIFS_CORES=128
        export FESOM="./fesom.x"
        export FESOM_CORES=144
        export RNFMAPPER="./rnfmap.exe"
        export RNFMAPPER_CORES=1

        export FESOM_USE_CPLNG="active"
        export ECE_CPL_NEMO_LIM="false"
        export ECE_CPL_FESOM_FESIM="false"
        export ECE_AWI_CPL_FESOM="true"

        # to address issue with srun
        COMPSS_MPI_TYPE=impi
        export
ECCODES_SAMPLES_PATH=/apps/ECCODES/2.8.0/INTEL/share/eccodes/ifs_samples/grib1/

        # Sample invocation of this script:
        # ./launch_mn4.sh 288 debug 1
        # ./launch_mn4.sh 144 debug 3
        # ./launch_mn4.sh 144 debug 1
        # $1 number of cores needed by the task
        # $2 QoS to be used (determines the queue)
        # $3 number of simulations of the ensemble

        EXP_ID=$(printf "%06d\n" $((1 + $RANDOM % 100000)))
        #EXP_ID="awi3"
        # prepare work folder
        mkdir $EXP_ID
        cd $EXP_ID
        cp -rf /home/bsc32/bsc32044/awicm3_ensemble/*.py  .
        cp -rf /home/bsc32/bsc32044/awicm3_ensemble/config .
        cp -rf /home/bsc32/bsc32044/awicm3_ensemble/hecuba_lib .

        # launch the esm ensemble simulation with hecuba infrastructure  through COMPSs
(WORKING)
        if [[ "${USE_HECUBA-}" == "TRUE" ]]; then
                echo "Running with HECUBA ENABLED"
                enqueue_compss -t -g -d --sc_cfg=mn.cfg  \
                       --qos=${QOS}  \
                       --storage_props=$PWD/hecuba_lib/storage_props.cfg \
                       --storage_home=$HECUBA_ROOT/compss \
                       --job_name=esm_workflow  \
                       --exec_time=120  \
                       --keep_workingdir \
                       --worker_working_dir=$PWD \
                       --worker_in_master_cpus=48  \
                       --num_nodes=${NODE_ALLOCATION}  \
                       --pythonpath=$PWD:$HECUBA_ROOT/compss esm_simulation.py ${EXP_ID}

        else
                echo "Running without HECUBA"
                enqueue_compss -t -g -d --sc_cfg=mn.cfg  \
                       --qos=${QOS}  \
                       --storage_props=$PWD/hecuba_lib/storage_props.cfg \
                       --job_name=esm_workflow  \
                       --exec_time=120  \
                       --keep_workingdir \
                       --worker_working_dir=$PWD \
                       --worker_in_master_cpus=48  \
                       --num_nodes=${NODE_ALLOCATION}  \
                       --pythonpath=$PWD esm_simulation.py ${EXP_ID}
```

Figure 2. ESM workflow initialization script.

The initialization of the ESM includes a PyCOMPSs configuration task that prepares the structure of working directories for each member of the ensemble, the namelists of the model for each ensemble member, and the output directories in the cluster. This can be seen in figure 2, near the comment “prepare work folder”, in bold. The Python scripts copied (*.py) contain more logic for handling the members and namelists.

The namelist files are generic and have meta-variables (also known as *placeholders*) that are replaced with the values defined in the master configuration file. This step must be performed before the model is executed. During the phase 3 of the iteration 2, this task was expanded to cover the coupling and OpenIFS settings.

4.2. ESM execution block

Figure 3 illustrates the main execution block for running an ensemble member of the model, with a periodic check of the state of the members to apply the pruning mechanism. The code was adapted to handle the results of the dynamic analysis (run in parallel), to support running OpenIFS and FESOM2 in coupled mode.

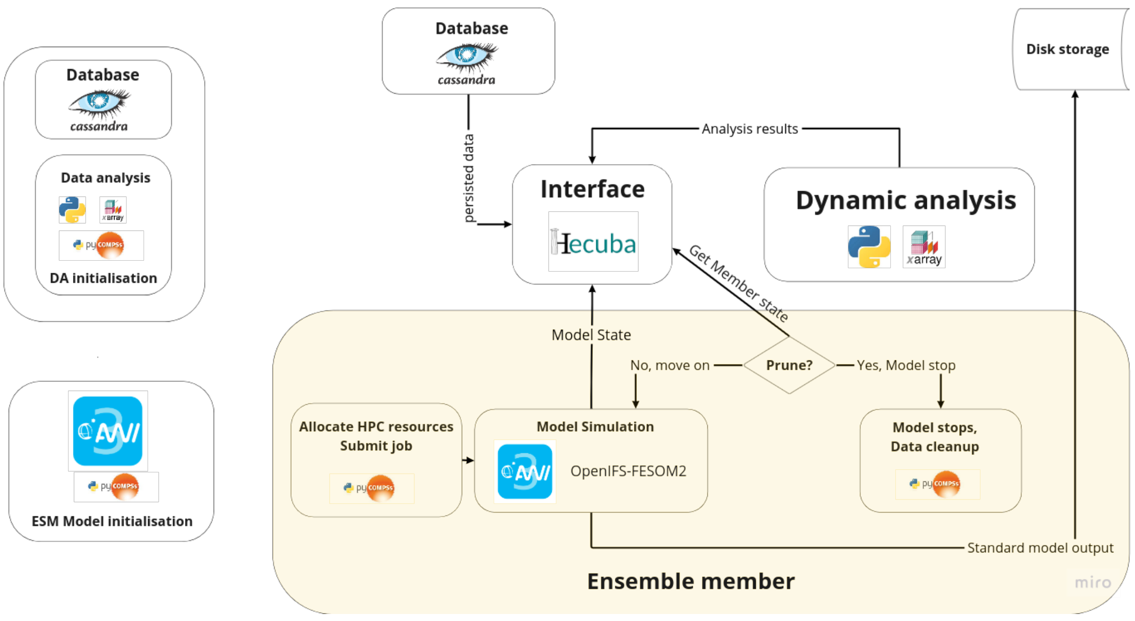


Figure 3. ESM dynamic workflow with the flow of data and events for the AWICM3 model.

The current implementation is able to run AWICM ensembles consisting of three start dates with TCO95L91-CORE2 configuration, using 336 cores distributed as follows:

- OIFS_CORES=128
- FESOM_CORES=144
- RNFMAPPER_CORES=1

The FESOM2 model was configured to use the COREII mesh, with a partition of 144 (distribution of the model mesh between CPUs).

```
@on_failure(management='IGNORE')
@mpmd_mpi(runner="srun", working_dir="{{working_dir_exe}}", fail_by_exit_value=True,
programs=[
dict(binary="./fesom.x", processes=144, args=""),
dict(binary="./master.exe", processes=128, args="-v ecmwf -e awi3"),
dict(binary="./rnfmap.exe", processes=1, args="")
])
@task(log_file={Type: FILE_OUT, StdIOStream: STDOUT}, working_dir_exe={Type: INOUT, Prefix: "#"},
to_continue={Type: IN, Prefix: "#"}, returns=int)
def esm_coupled_simulation(log_file, working_dir_exe, to_continue):
pass
```

Figure 4. PyCOMPSs code to execute AWICM3 simulations.

The complete code is located on GitHub repository, at: https://earth.bsc.es/gitlab/ces/eflows4hpc-wp5/

4.3. Integration with HPCWaaS

Another achievement was the creation of an application in Alien4Cloud to execute the ESM workflow. This involved modular containerization of workflow components and orchestrating them on HPC using a web interface (e.g, figure 5).

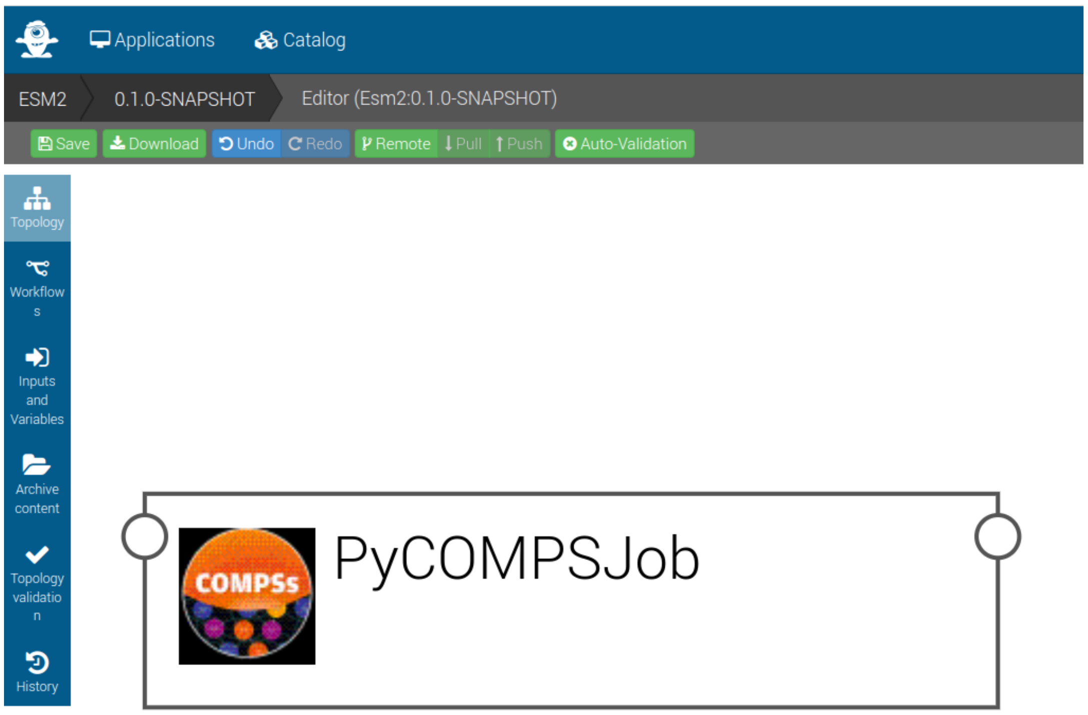


Figure 5. PyCOMPSsJOB Tosca template in Alien4Cloud.

A Tosca template shown in figure 6 is used to execute the application on MareNostrum 4.

```
         tosca_definitions_version: alien_dsl_3_0_0

         metadata:
           template_name: Esm2
           template_version: 0.1.0-ESMDW_Coupled-SNAPSHOT
           template_author: julian

         description: ""

         imports:
           - yorc-types:1.1.0
           - tosca-normative-types:1.0.0-ALIEN20
           - alien-base-types:3.0.0
           - pycomps.ansible:1.2.0-SNAPSHOT
           - dls.ansible:1.1.0-SNAPSHOT

         topology_template:
          inputs:
             debug:
               type: boolean
               required: true
               default: false
               description: "Do not redact sensible information on logs"
             target_host:
               type: string
               required: true
               description: "the remote host"
             user_id:
               type: string
               required: false
               default: ""
               description: "User id to use for authentication may be replaced with workflow input"
             vault_id:
               type: string
               required: false
               default: ""
               description: "Vault id to use for authentication may be replaced with workflow input"
           node_templates:
             ESM_Workflow:
               metadata:
                 a4c_edit_x: 5
                 a4c_edit_y: "-46"
               type: pycomps.ansible.nodes.PyCOMPSJob
               properties:
                 pycomps_endpoint: { get_input: target_host }
                 compss_module_version: eflows4hpc
                 num_nodes: 7
                 qos: debug
                 input_data_path: "/home/bsc32/bsc32044/awicm3_ensemble_alien4cloud/input"
                 output_data_path: "/home/bsc32/bsc32044/awicm3_ensemble_alien4cloud/output"
                 command: "/home/bsc32/bsc32044/awicm3_ensemble_alien4cloud/esm_simulation.py"
                 arguments:
                   - a0000001
                 container_image: ""
                 container_compss_path: ""
                 container_opts: ""
                 python_interpreter: python3
                 extra_compss_opts:         "--qos=debug         --exec_time=120         --keep_workingdir         --
worker_working_dir=/home/bsc32/bsc32044/awicm3_ensemble_alien4cloud    --worker_in_master_cpus=48    --num_nodes=7    --
pythonpath=/home/bsc32/bsc32044/awicm3_ensemble_alien4cloud:/apps/HECUBA/2.0/compss                                  --
env_script=/home/bsc32/bsc32044/awicm3_ensemble_alien4cloud/set_esm_workflow_env.sh                                  --
storage_props=/home/bsc32/bsc32044/awicm3_ensemble_alien4cloud/hecuba_lib/storage_props.cfg                          --
storage_home=/apps/HECUBA/2.0/compss"
           workflows:
             exec_job:
               inputs:
                 user_id:
                   type: string
                   required: true
                 vault_id:
                   type: string
                   required: true
                 target_path:
                   type: string
                   required: true
                 source_path:
                   type: string
                   required: true
                 num_nodes:
                   type: integer
                   required: false
                   default: 1
               steps:
                 PyCOMPSJob_submitting:
                   target: ESM_Workflow
                   activities:
                     - set_state: submitting
                   on_success:
                     - PyCOMPSJob_submit
                 PyCOMPSJob_submit:
```

```
  target: ESM_Workflow
  operation_host: ORCHESTRATOR
  activities:
    - call_operation: tosca.interfaces.node.lifecycle.Runnable.submit
  on_success:
    - PyCOMPSJob_submitted
PyCOMPSJob_executing:
  target: ESM_Workflow
  activities:
    - set_state: executing
  on_success:
    - PyCOMPSJob_run
PyCOMPSJob_executed:
  target: ESM_Workflow
  activities:
    - set_state: executed
PyCOMPSJob_submitted:
  target: ESM_Workflow
  activities:
    - set_state: submitted
  on_success:
    - PyCOMPSJob_executing
PyCOMPSJob_run:
  target: ESM_Workflow
  operation_host: ORCHESTRATOR
  activities:
    - call_operation: tosca.interfaces.node.lifecycle.Runnable.run
  on_success:
    - PyCOMPSJob_executed
```

Figure 6. Tosca code for running the ESM dynamic workflow.

This ESM dynamic workflow application is deployed and executed by using Alien4Cloud. Currently, it relies on files and directories that are hard-coded in the Tosca definition. For the next phase, these values will be parametrized, making it easier to share and re-use the application. The workflow additionally needs to implement a data logistics component to manage the input and outputs of model simulation for modularity.

This application can then be deployed to any instance of Alien4Cloud. Authentication and authorization to execute a workflow on a HPC are handled by Alien4Cloud and it also uses a vault for credentials such as SSH keys used to safely communicate with the HPC platforms. Users are asked to provide a vault_id that corresponds to a secure SSH key to be used to run the ESM.

4.4. Next steps towards the end of the project

For the next and final phase of the project, we would like to accomplish the following list of tasks:

- Consolidate the workflows by parameterizing the Tosca definition to promote re-use and portability of the ESM workflow application.
- Add a workflow component to disseminate simulation data from HPC to public cloud storage service or DLS developed in WP2.
- Benchmark the throughput of model data to Cassandra database via Hecuba and prepare manuscript for publication.
- Refine the pruning of ensemble members for model tuning exercise of FESOM2, present the results at American Geophysical Union Fall Meeting 2023 and prepare the manuscript for publication.
- Propose a session on HPC workflows for upcoming European Geophysical Union General Assembly 2024 to bring community working on workflows and share insights.

5. Feature extraction workflow

The statistical analysis and feature extraction workflow of Pillar II integrates the ESM model execution together with HPDA and ML approaches for the analysis of extreme events (e.g., Heat/Cold Waves and Tropical Cyclones) based on the output of the ESM model simulation.

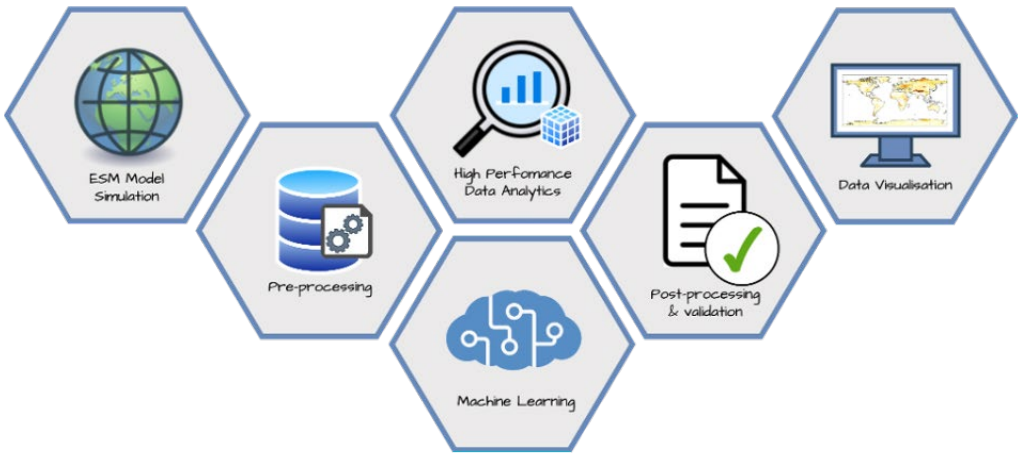


Figure 7. General overview of the feature extraction workflow.

The figure 7 shows a high-level overview of the main blocks integrated in the whole workflow that have been analyzed and described in the deliverable “D5.3 Pillar II - Iteration 1 Software Release”.

This section provides a general outlook of the current state of implementation, and then describes the most relevant implementation details of the building blocks that have been developed during this phase.

5.1. Detailed description of the workflow

The schematization of three implementation use cases has been provided in “D5.3 Pillar II - Iteration 1 Software Release”, where two main potential implementation use cases were defined. The first use case runs the feature extraction analysis (i.e., TC detection and Heat Waves analysis) on the CMCC-CM3 model data, already fully available on disk, while in the second all the pipelines (Deterministic and ML TC Detection blocks and Analytics block) are executed together with the ESM simulation.

Since the second use case represents a more complete and complex scenario, it has included the first one, as shown in the following figure.

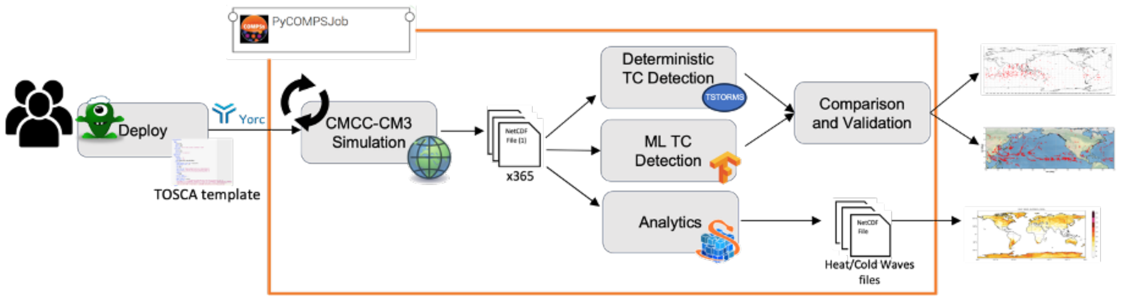


Figure 8. High-level overview of the workflow for extreme climate events use case. The first use case is included in this one, starting from the 365 files produced by the model to the maps production.

The Pillar II statistical analysis and feature extraction workflow is organized in three main stages:

- The deployment of the high-level workflow using Alien4Cloud and YORC
- The management of the tasks by PyCOMPSs
- The visualization of the results (NetCDF[13] files, maps, images).

The PyCOMPSs job includes the following steps:

1. Execution of the CMCC-CM3 model simulation
2. As soon as the ESM output data becomes available, the data analytics and ML tasks run on the new yearly variables produced by the model:
    a. The HPDA-based extreme events analysis (i.e., Heat Waves and Cold Spells)
    b. The deterministic-based TC detection
    c. The ML-based TC detection
3. The output of the analysis is then validated and stored on the disk
4. Production of the output maps on the post-processing results.

At runtime, PyCOMPSs generates a task-dependency graph as shown in figure 9 by analyzing the existing data dependencies between the tasks defined in the Python code.

For the sake of clarity, the graph shows the execution of the workflow for a single year of simulated data; in case of multiple years, the whole graph will be repeated (with the exception of the first four tasks). More in detail, the figure shows the list of the PyCOMPSs routines declared in different python scripts in the modules folder:

- The RunCMCCModel (blue circle) is the task responsible for the CMCC-CM3 simulation execution;
- The OphImportClimAvg task (white circles) corresponds to the import of the climatological mean NetCDF file into Ophidia, while the “Stream4” associated to the red circle is useful to synchronize the following tasks that start only when the CMCC-CM3 data are available on disk;
- The tasks from 5 to 14 are related to the Ophidia operators necessary to compute the Heat/Cold Wave Indicators;
- The ML tasks (ComputePatches, InferencePhase and CreatePrediction) deal with the preprocessing step, detecting TC presence and localizing the TC center position;
- The last three tasks (18, 19, 20) are responsible for the execution of the TSTORMS that starts only when the preprocessing step that aggregates the daily files into monthly files is finished thanks to the “Stream80” interface.

All tasks along the X axis are executed in parallel and the “sync” blocks correspond to explicit synchronizations, while the other tasks have implicit synchronizations inside.

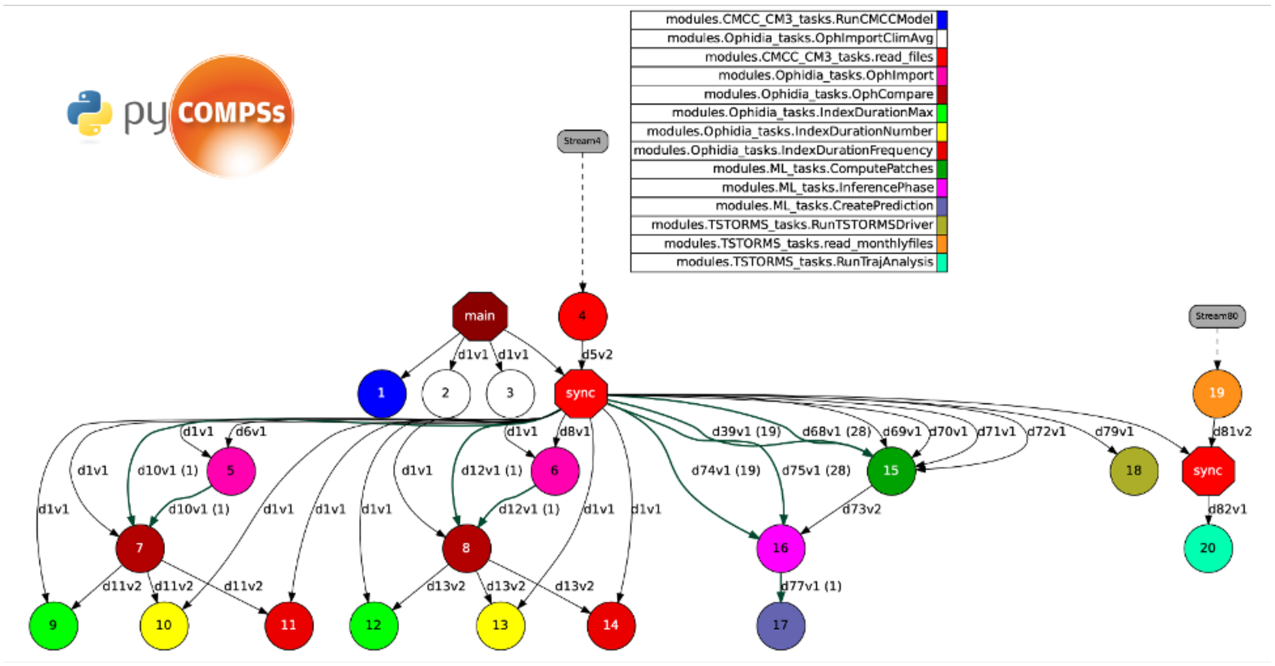
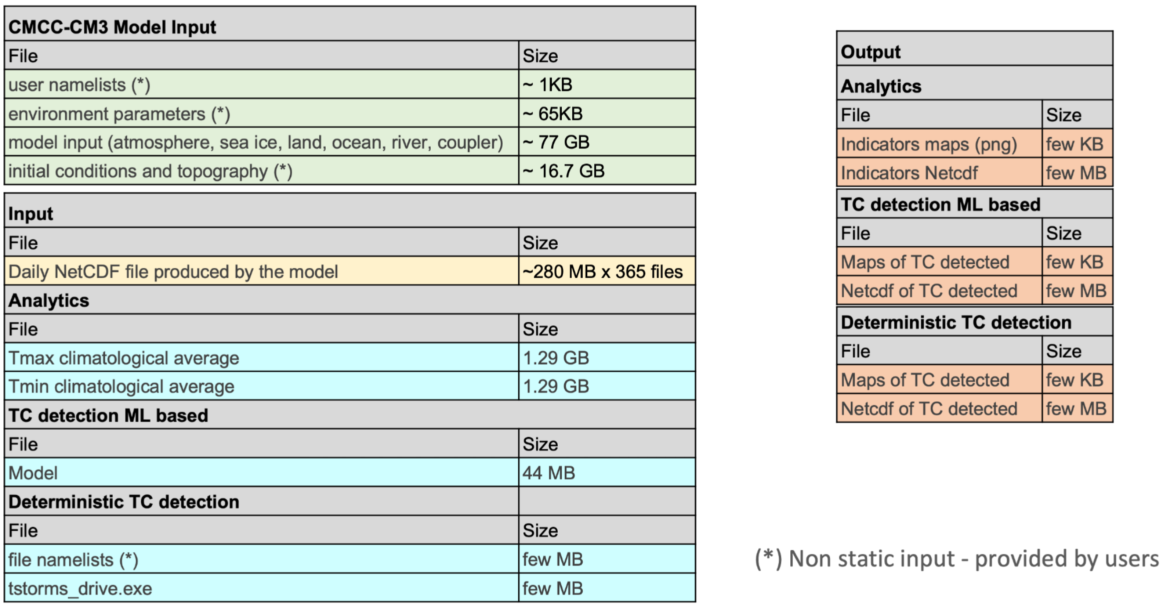

py COMPSs
modules.CMCC_CM3_tasks.RunCMCCModel
modules.Ophidia_tasks.OphImportClimAvg
modules.CMCC_CM3_tasks.read_files
modules.Ophidia_tasks.OphImport
modules.Ophidia_tasks.OphCompare
modules.Ophidia_tasks.IndexDurationMax
modules.Ophidia_tasks.IndexDurationNumber
modules.Ophidia_tasks.IndexDurationFrequency
modules.ML_tasks.ComputePatches
modules.ML_tasks.InferencePhase
modules.ML_tasks.CreatePrediction
modules.TSTORMS_tasks.RunTSTORMSDriver
modules.TSTORMS_tasks.read_monthlyfiles
modules.TSTORMS_tasks.RunTrajAnalysis
Stream4
Stream80
main
sync
sync

Figure 9. PyCOMPSs task-dependency graph for feature extraction workflow.

The necessary inputs and outputs for the entire workflow have been summarized in the tables below in order to identify the blocks that can be portable to different computing infrastructures and that need some input files to be performed.

It is important to underline that the CMCC-CM3 model runs on the CMCC supercomputer Zeus and is not portable to other infrastructures, so the idea is to separate the workflow into two sub-workflows:

- CMCC-CM3 model simulation: runs on Zeus and produce the daily files;
- Post processing steps: run anywhere starting from the files produced by the model and made available on other computing infrastructures.



CMCC-CM3 Model Input	
File	Size
user namelists (*)	~ 1KB
environment parameters (*)	~ 65KB
model input (atmosphere, sea ice, land, ocean, river, coupler)	~ 77 GB
initial conditions and topography (*)	~ 16.7 GB

Input	
File	Size
Daily NetCDF file produced by the model	~280 MB x 365 files
Analytics	
File	Size
Tmax climatological average	1.29 GB
Tmin climatological average	1.29 GB
TC detection ML based	
File	Size
Model	44 MB
Deterministic TC detection	
File	Size
file namelists (*)	few MB
tstorms_drive.exe	few MB

Output	
Analytics	
File	Size
Indicators maps (png)	few KB
Indicators Netcdf	few MB
TC detection ML based	
File	Size
Maps of TC detected	few KB
Netcdf of TC detected	few MB
Deterministic TC detection	
File	Size
Maps of TC detected	few KB
Netcdf of TC detected	few MB

(*) Non static input - provided by users

Figure 10. Necessary inputs and outputs for the entire workflow.

The following subsections describe all the workflow stages in detail, along with the related PyCOMPSs tasks.

5.2. ESM simulation block

In the ESM simulation block the CMCC-CM3 climate model is used, as reported in the deliverable D5.3 "D5.3 Pillar II - Iteration 1 Software Release", to produce the necessary files for the subsequent blocks. It represents task 1 in the PyCOMPSs graph. The main change made in this phase of the project concerns the use of the STREAM IN parameter for files implemented in the read_files task (red circle in the figure). This option allows to detect the files production progress and to poll the stream data from the other tasks. It is very useful because at the end of the simulation, all the daily CMCC-CM3 output files are copied to a directory that will be monitored using the PyCOMPSs streaming interface.

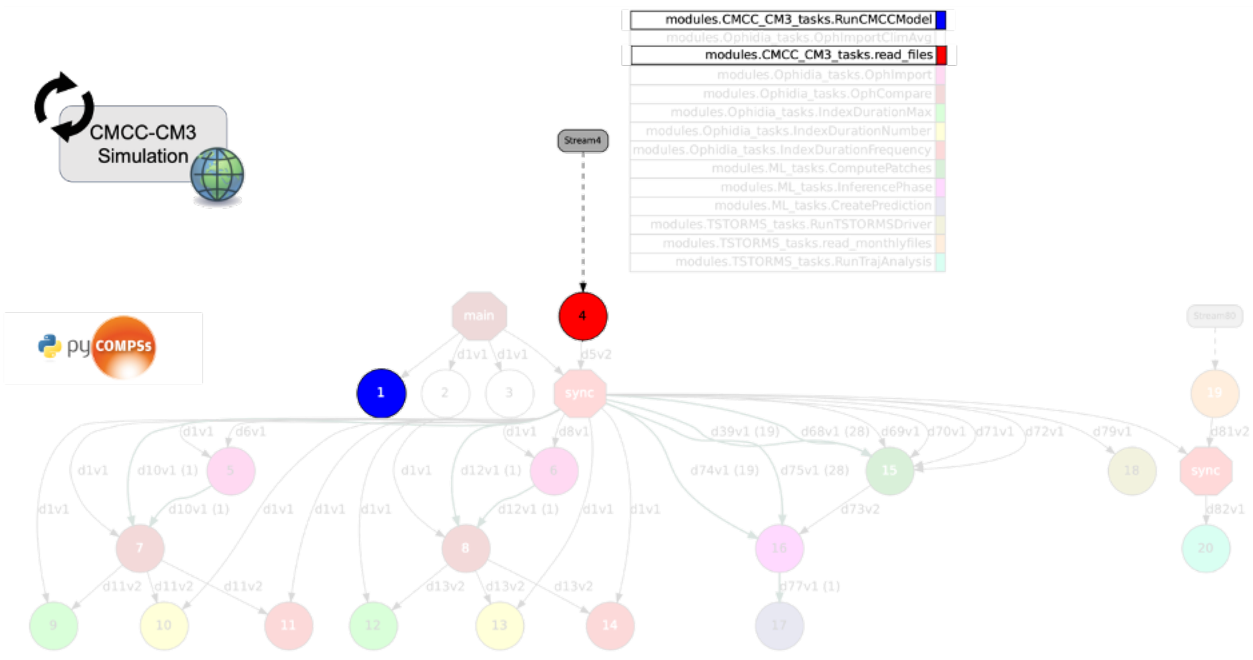

CMCC-CM3
Simulation
py COMPSs
modules.CMCC_CM3_tasks.RunCMCCModel
modules.Ophidia_tasks.OphImportClimAvg
modules.CMCC_CM3_tasks.read_files
modules.Ophidia_tasks.OphImport
modules.Ophidia_tasks.OphCompare
modules.Ophidia_tasks.IndexDurationMax
modules.Ophidia_tasks.IndexDurationNumber
modules.Ophidia_tasks.IndexDurationFrequency
modules.ML_tasks.ComputePatches
modules.ML_tasks.InferencePhase
modules.ML_tasks.CreatePrediction
modules.TSTORMS_tasks.RunTSTORMSDriver
modules.TSTORMS_tasks.read_monthlyfiles
modules.TSTORMS_tasks.RunTrajAnalysis
Stream4
main
4
1
2
3
sync
d5v2
19
sync
20
15
16
17
18
5
6
7
8
9
10
11
12
13
14

Figure 11. PyCOMPSs task-dependency graph focused on CMCC-CM3 Simulation.

The definition of the *read_files* task is shown in the code below in which the list_files array represents the list of all daily files making up the new year.

```
@task(fds=STREAM_IN, returns=list)
def read_files(fds):
    num_total = 0
    list_files = []
    while num_total < 365:
        # Poll new files
        new_files = fds.poll()
        # Process files
        for nf in new_files:
            list_files.append(str(nf))
        # Accumulate read files
        num_total = len(list_files)
        # Sleep between requests
        time.sleep(SLEEP)
    # Return the number of processed files
    return list_files
```

5.3. Data analytics for extreme events analysis

The data analytics block of the workflow is responsible for the computation of multiple extreme climate events indicators exploiting the Ophidia framework for HPDA. The multiple indicators computed in this stage for Heat Waves and Cold Spells are summarized in the following table taken from the deliverable "D5.3 Pillar II - Iteration 1 Software Release".

Table 1. List of extreme events indices computed by the workflow.

Index	Name	Description
HWD	Heat Wave Duration	Starting from the daily maximum temperature (**TSMX**), the Heat Wave Duration index is the maximum number of days at intervals of at least 6 days with TSMX > 5°C + baseline **BASELINE**: Average calculated for each calendar day (based on 20 years) using a current 5-day window
CWD	Cold Wave Duration	Starting from the daily minimum temperature (**TSMN**), the Cold Wave Duration index is the maximum number of days at intervals of at least 6 days with TSMN < 5°C + baseline
HWN	Heat Wave Number	Number of heatwaves in a year
CWN	Cold Wave Number	Number of coldwaves in a year
HWF	Heat Wave Frequency	Number of days that contribute to heatwaves in a year
CWF	Cold Wave Frequency	Number of days that contribute to coldwaves in a year

To compute these indicators, the variables TSMX (Maximum surface temperature) and TSMN (Minimum surface temperature) are used.

Figure 12 provides a schematic representation of the internal steps computed in this block. The main change made in this phase of the project involves the use of the PyCOMPSs API function *compss_file_exists(*file_name)* that is able to check if a file or files exist. This mechanism allows to simplify the procedure, because if the climatological mean NetCDF file is available, the green step in the figure for the Climatological mean computation will be reduced in a simple import into Ophidia of this file.

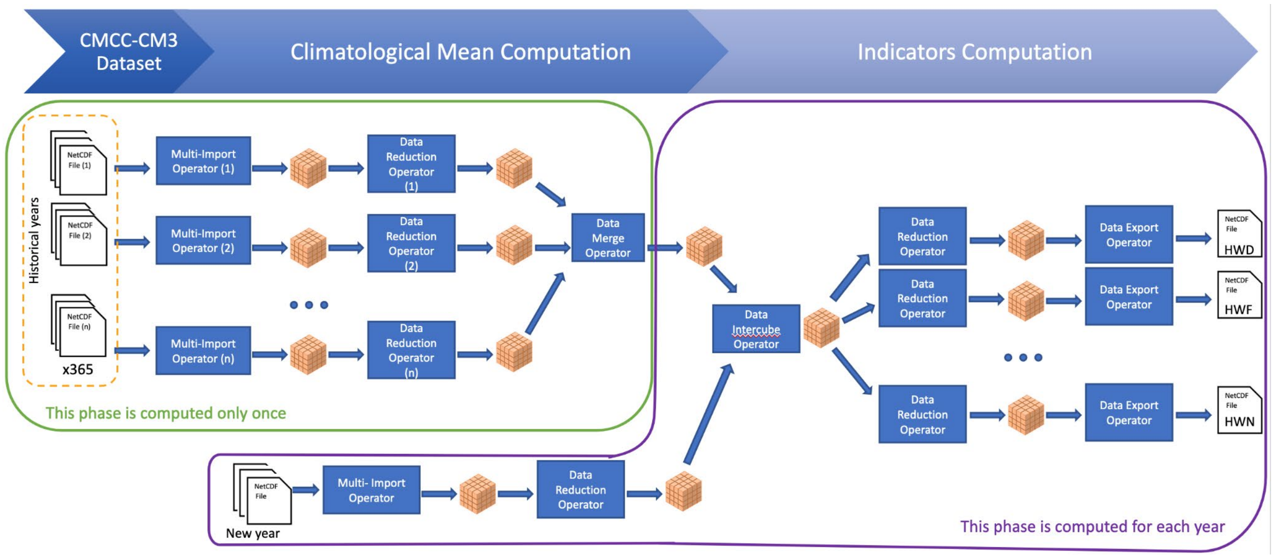

CMCC-CM3 Dataset
Climatological Mean Computation
Indicators Computation
Historical years
NetCDF File (1)
NetCDF File (2)
NetCDF File (n)
x365
Multi-Import Operator (1)
Multi-Import Operator (2)
Multi-Import Operator (n)
Data Reduction Operator (1)
Data Reduction Operator (2)
Data Reduction Operator (n)
Data Merge Operator
This phase is computed only once
Data Intercube Operator
Data Reduction Operator
Data Export Operator
HWD
HWF
HWN
New year
Multi- Import Operator
This phase is computed for each year

Figure 12. Internal steps for the HPDA-based extreme events analysis block.

As soon as a new file is produced from the ESM simulation, the Heat Waves and Cold Spell events are extracted from the files making use of some Ophidia operators (for reduction, intercomparison, etc.).

PyCOMPSs is used to perform the Ophidia pipelines concurrently on different input files and orchestrates the execution of the various operators, as shown in the graph below.

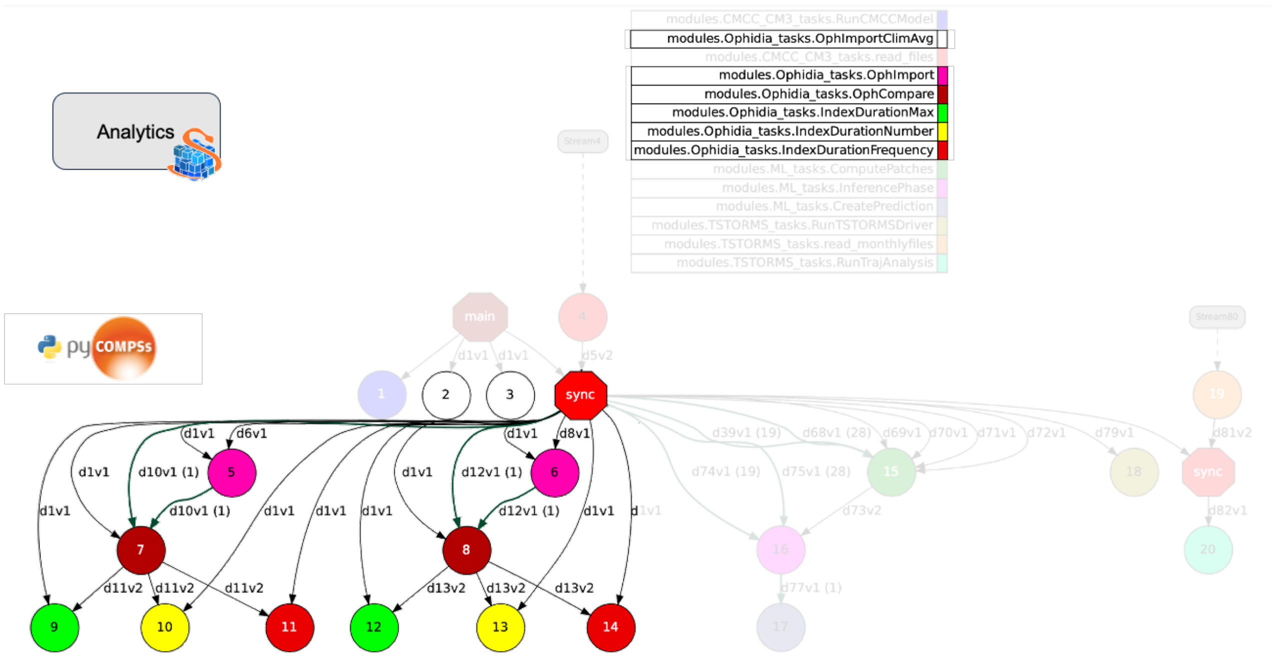

Analytics
py COMPSs
modules.Ophidia_tasks.OphImportClimAvg
modules.Ophidia_tasks.OphImport
modules.Ophidia_tasks.OphCompare
modules.Ophidia_tasks.IndexDurationMax
modules.Ophidia_tasks.IndexDurationNumber
modules.Ophidia_tasks.IndexDurationFrequency
main
sync
1
2
3
5
6
7
8
9
10
11
12
13
14

Figure 13. PyCOMPSs task-dependency graph focused on Data Analytics.

5.4. Detecting Tropical Cyclones in ESMs simulations

The Machine Learning workflow extracts significant spatial features related to the presence of TCs in gridded climate data. This allows recognition, in an end-to-end fashion, of whether a TC is present or not given a set of input climate variables that are related to TCs cyclogenesis and sustainment during their lifecycle. From a logical point of view the TC detection workflow is based on two subsequent steps: *(i)* classification and *(ii)* localization.

The former allows classification of whether a TC is likely to occur given input data, whereas the latter allows one to effectively localize the geographical coordinates of the TC. Practically, both classification and localization steps are embedded into a comprehensive end-to-end detection task which has been addressed through the use of VGG-like Artificial Neural Network (ANNs) architectures. VGGs exploit deep convolutions to extract high-level spatial features from multi-dimensional gridded data with the aim of predicting, when a TC has been detected, the geographical coordinates of the TC center.

The ML-enabled TCs detection workflow is made up of two distinct phases: training and inference. While the training phase trains VGGs for detection on historical reanalysis data (i.e., ERA5), the inference phase exploits pre-trained VGGs to infer the presence or absence of TCs in global model simulation data (i.e., CMCC-CM3), thus providing an indication of their global geographical occurrence in the future.

5.4.1. Training phase

As described in D5.3, the ML training workflow for TCs detection comprises three consecutive stages. Stage 1 involves the collection of all data sources required to set up the ML models training for the detection task. In this stage some pre-processing steps are also performed, such as patch generation and labeling, augmentation (i.e., left-right flip, up-down flip and 180° rotation) and feature scaling. Stages 2 and 3 are related to classification and localization tasks, respectively. As previously described, these stages are logically separated but from a ML perspective they are jointly performed in an end-to-end way. Indeed, VGG models are trained to predict the geographical coordinates of the TC center if it is detected in input patches, otherwise negative labels are provided as output. From ERA5, 4 input drivers related to the Tropical Cyclone (TC) formation have been identified: 10m wind gust since previous post-processing (fg10), temperature at 500 mb (t_500), temperature at 300 mb (t_300) and mean sea level pressure (msl). These variables were gathered from ERA5 single levels and ERA5 pressure levels datasets (0.25° x 0.25° on regular grid). Information about TC position and development have been gathered from the International Best Track Archive for Climate Stewardship (IBTrACS).

The main contributions with respect to D5.3 are:

- Improved IBTrACS filtering: selection of 6-hourly records belonging to Tropical Storm (TS), Extratropical (ET) and Subtropical (SS) categories occurring in the joint North Pacific and North Atlantic formation basins (100−320 °E, 0−70 °N);
- Data is converted from NetCDF into a TensorFlow[14] compliant data format (i.e., tfrecord) which allows faster I/O operations, as well as enabling faster training on GPUs. Indeed, tfrecords allow improving the training execution time by a factor of 3 over using Numpy[15] arrays (about 3 hours to train a VGG-line ANN with tfrecords);
- Training and Validation datasets now include:
    - Patches containing a TC associated with the corresponding coordinates of the TC center (positive patches);
    - For each positive patch, the three corner patches closest to the storm center were considered and labeled with negative coordinates (i.e., [-1,-1]) (negative patches);

  - For each positive patch, a further negative patch was randomly selected among the 7 × 22 patches of the map excluding the previously mentioned, additionally ensuring that no major TC phenomena occur in the randomly selected patch.
- Improved augmentation of gridded data: training and validation patches undergo data augmentation (i.e., left-right flip, up-down flip and 180° rotation) which is performed on the fly, avoiding storing the augmented version of data on the storage system;
- New evaluation metrics: performance on out-of-sample data is evaluated by using classification metrics, such as False Positives (FPs), False Negatives (FNs), True Positives (TPs) and True Negatives (TNs), as well as localization metrics, such as the average Euclidean distance between correctly identified TCs (TPs) and observations from IBTrACS;
- A total of 13 models have been trained and fine-tuned for the detection task on historical data, as well as their classification and localization metrics on the test set (see Table 2).
  - A lower number of FNs is preferable for the task of predicting the occurrence of such extreme events, because FNs correspond to observed TCs that are not cor-rectly identified by the ML model;
  - However, the Euclidean distance between the observed and predicted TCs center is in trade-off with respect to FNs, the higher the former the lower the latter and vice versa;
  - To this extent a total of 13 models have been trained and tested, each one with a different balance between Euclidean Distance and FN rate. Model #10 should be se-lected for an accurate localization of TCs center (112.81 km on average), whereas model #7 for minimizing the occurrence of FNs (9.13 % on average) at the expense of the localization accuracy which increases to 190.16 km. A bal-anced behavior is provided by model #5 with an average Euclidean Distance of 143.59 km which is about lower by 50 km than model #7, at the expense of the FN rate which increases from 9.13 % (model #7) to 13.60 %.

Table 2. List of trained VGG-line ANNs on historical data. The average Euclidean distance between predicted and actual TC center coordinated on the test set was reported, along with classification rates.

Model #	**Euclidean distance (km)**	**FP rate (%)**	**TP rate (%)**	**FN rate (%)**	**TN rate (%)**
1	143.48	7.61	85.27	14.73	92.39
2	188.94	23.00	90.78	9.22	77.00
3	190.66	17.93	90.81	9.19	82.07
4	124.96	3.78	73.36	26.64	96.22
5	143.59	7.70	86.40	13.60	92.30
6	193.02	16.28	90.23	9.77	83.72
7	190.16	21.32	90.87	9.13	78.68
8	129.88	4.53	74.84	25.16	95.47

9	133.48	3.30	77.26	22.74	96.70
10	112.81	3.46	77.46	22.54	96.54
11	114.41	3.55	76.34	23.66	96.45
12	114.13	3.74	75.27	24.73	96.26
13	115.47	3.73	73.68	26.32	96.27

5.4.2. Inference phase

ML model inference is performed on climate projections provided by the CMCC-CM3 numerical model. To this end, 6-hourly projection data has been collected and used. As a preliminary step, the correspondence between ERA5 variables, used in the training phase, and those simulated by the CMCC-CM3 numerical model was established. This correspondence is reported in Table 3.

Table 3. Correspondence between ERA5 variables used in the training phase and those simulated by the CMCC-CM3 numerical model.

Variable Name	**Unit**	**ERA5 Name**	**CMCC-CM3 Name**
10 m wind gust since previous post-processing	m/s	fg10	WSPDSRFMX
temperature at 500 mb	K	t_500	T500
temperature at 300 mb	K	t_300	T300
mean sea level pressure	Pa	msl	PSL

CMCC-CM3 data has been regridded onto a 0.25° x 0.25° grid to match the same spatial resolution of ERA5 data. Then, each map was tiled into non-overlapping patches of size 40 x 40 pixels. Variable patches were then stacked together, resulting in objects of size 40 x 40 x 4. Then, patches were normalized in the [0,1] range by using the same scaler computed on the training set. Such data is then fed into the ML model to detect TCs on simulation data.

Figure 14 shows the PyCOMPSs task-dependency graph for the ML-enabled TC detection.

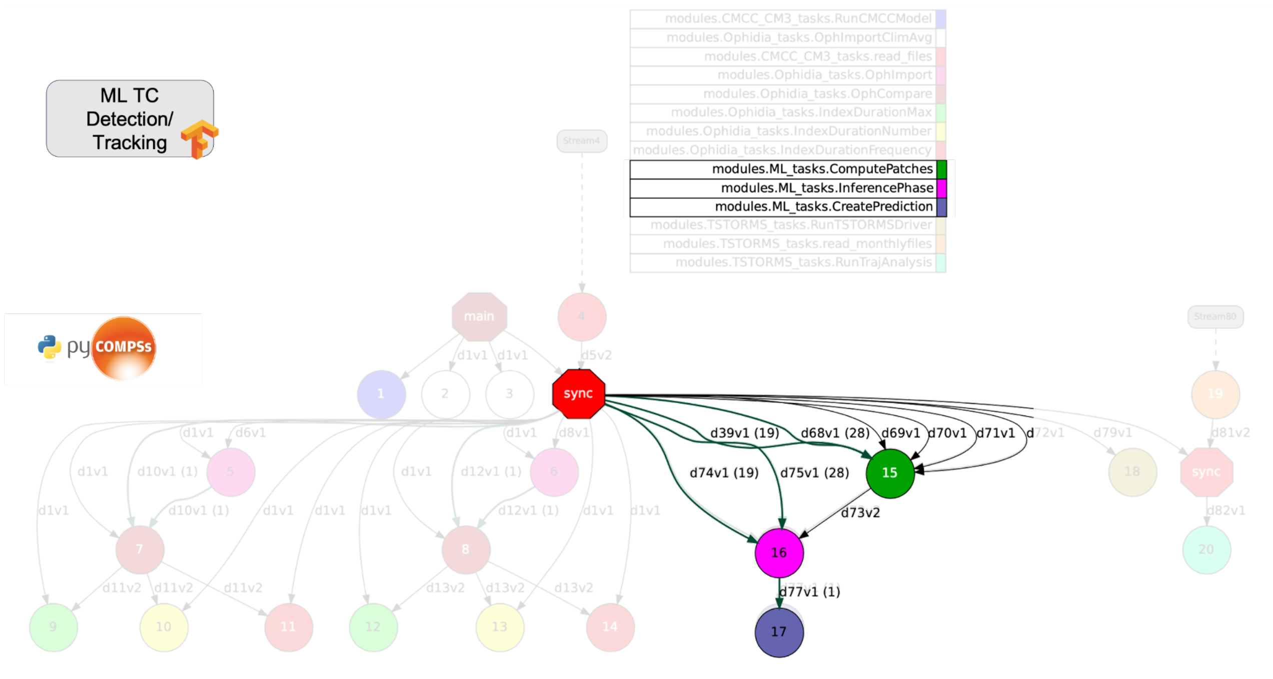

ML TC Detection/ Tracking
modules.CMCC_CM3_tasks.RunCMCCModel
modules.Ophidia_tasks.OphImportClimAvg
modules.CMCC_CM3_tasks.read_files
modules.Ophidia_tasks.OphImport
modules.Ophidia_tasks.OphCompare
modules.Ophidia_tasks.IndexDurationMax
modules.Ophidia_tasks.IndexDurationNumber
modules.Ophidia_tasks.IndexDurationFrequency
modules.ML_tasks.ComputePatches
modules.ML_tasks.InferencePhase
modules.ML_tasks.CreatePrediction
modules.TSTORMS_tasks.RunTSTORMSDriver
modules.TSTORMS_tasks.read_monthlyfiles
modules.TSTORMS_tasks.RunTrajAnalysis
py COMPSs
sync
d39v1 (19)
d68v1 (28)
d69v1
d70v1
d71v1
d74v1 (19)
d75v1 (28)
d73v2
d77v1 (1)
15
16
17

Figure 14. PyCOMPSs task-dependency graph focused on ML TC detection.

5.4.3. Deterministic TC detection

The integration of the TSTORMS tool for the deterministic TC detection in the overall PyCOMPSs workflow represents one of the most relevant implementation advances that have been developed during this phase.

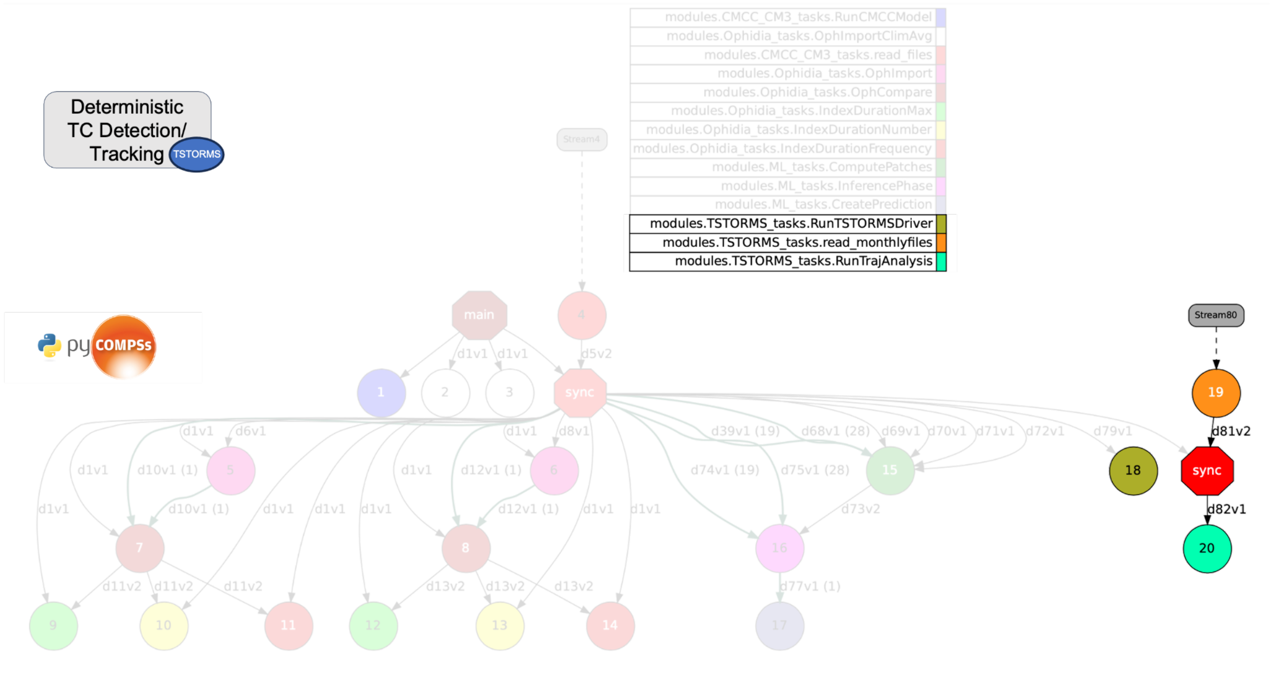

Deterministic TC Detection/ Tracking
TSTORMS
modules.CMCC_CM3_tasks.RunCMCCModel
modules.Ophidia_tasks.OphImportClimAvg
modules.CMCC_CM3_tasks.read_files
modules.Ophidia_tasks.OphImport
modules.Ophidia_tasks.OphCompare
modules.Ophidia_tasks.IndexDurationMax
modules.Ophidia_tasks.IndexDurationNumber
modules.Ophidia_tasks.IndexDurationFrequency
modules.ML_tasks.ComputePatches
modules.ML_tasks.InferencePhase
modules.ML_tasks.CreatePrediction
modules.TSTORMS_tasks.RunTSTORMSDriver
modules.TSTORMS_tasks.read_monthlyfiles
modules.TSTORMS_tasks.RunTrajAnalysis
py COMPSs
Stream80
19
d81v2
18
sync
d82v1
20

Figure 15. PyCOMPSs task-dependency graph focused on TSTORMS TC detection.

As shown in the figure 15, the TC detection via a deterministic approach is integrated using two PyCOMPSs tasks:

- The first *RunTSTORMSDriver* starts only if the CMCC-CM3 model has produced the new daily files. It is useful to aggregate the daily files in order to produce monthly files and execute the TSTORMS driver using them to detect the tropical cyclones points;
- The second one, *RunTrajAnalysis,* is responsible for the trajectory analysis to keep the information on all years. Basically, this task joins the cyclones points.

The STREAM_IN option has been used also in this case (orange circle) to wait on the TSTORMS files creation before running the trajectory analysis task that uses these files.

The following code is related to the declaration and the invocation of the *RunTSTORMSDriver* task and the *read_monthlyfiles* task that implements the PyCOMPSs streaming interface.

```
@binary(binary="runTSTORMS.py")
@task(files=COLLECTION_IN)
def RunTSTORMSDriver(str_files, files):
    pass
# TSTORMS execution
        RunTSTORMSDriver(str(listfiles),listfiles)
        print("[LOG] TSTORMS EXECUTION")
        tstormsfiles = read_monthlyfiles(fds1,year)
        # Sync and print value
        tstormsfiles = compss_wait_on(tstormsfiles)
        print("[LOG] PROCESSED TSTORMS FILES: " + str(tstormsfiles))
```

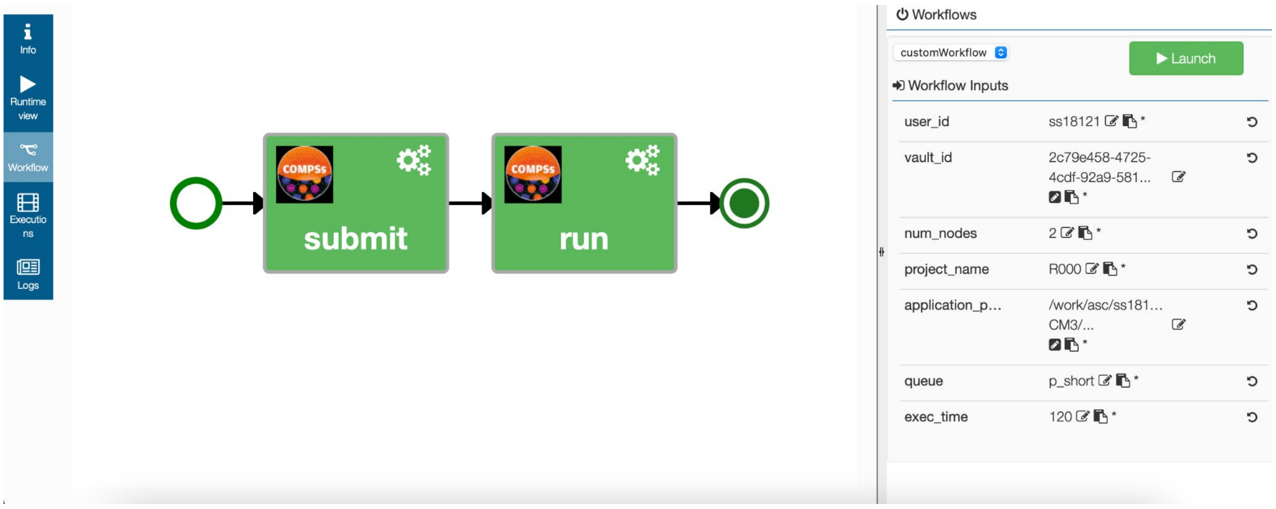
5.5. Testbed setup

Zeus is one of the HPC machines available at the CMCC supercomputing center[1]. It provides 1.2 PetaFlops of peak performance and is composed of 348 nodes with a total of 12,528 processors and 33.4 TB of main memory. Each node is equipped with 2 Intel Xeon Gold 6154 3.0 GHz processors (18 cores each) and 96 GB of main memory, with Linux CentOS 7.6 used as the operating system. The cluster exploits a GPFS parallel file system with a peak aggregated bandwidth of 80 GB/s and IBM Spectrum LSF as scheduling system.

5.6. Integration with HPCWaaS

Alien4Cloud is a tool that provides self-service deployment, composition and execution of complex applications by means of application architectures described in the TOSCA Simple Profile in YAML specification.

[1] CMCC SuperComputing Center: https://www.cmcc.it/super-computing-center-scc



user_id	ss18121 *
vault_id	2c79e458-4725-4cdf-92a9-581... *
num_nodes	2 *
project_name	R000 *
application_p...	/work/asc/ss181... CM3/... *
queue	p_short *
exec_time	120 *

Figure 16. PyCOMPSs workflow architecture in Alien4Cloud.

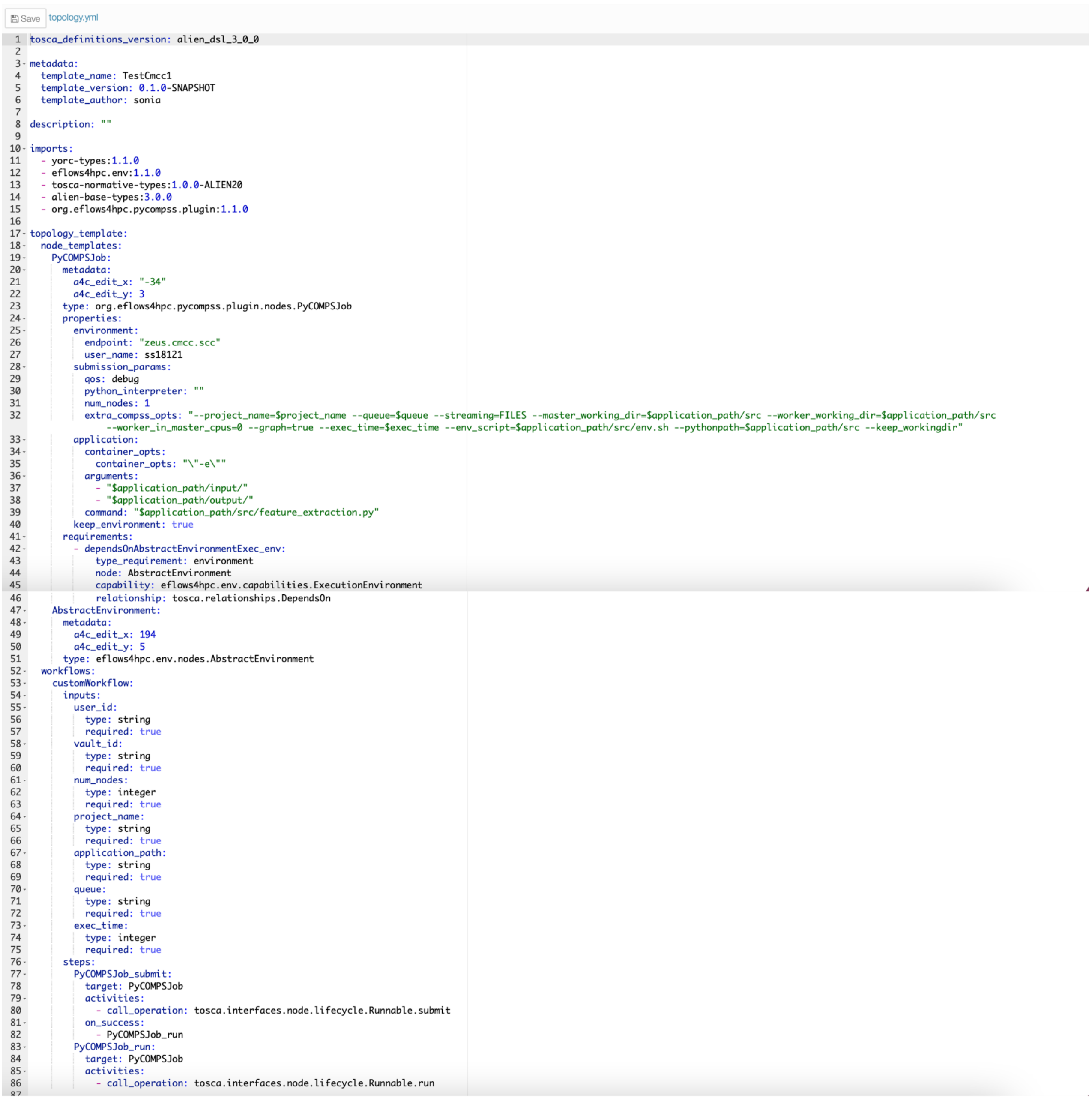
The figure 17 shows the definition of the Tosca configuration file for the YORC technology to support the deployment of the workflow and its execution on different infrastructures.

The main parameters concern:

- the environment endpoint corresponding to the Zeus supercomputer so a VPN connection has been established between Zeus at CMCC and the machine hosting Alien4Cloud.
- the extra_compss_opts in the submission params in which are defined all the options needed by the command enqueue_compss to execute the PyCOMPSs workflow.
- the application arguments in which are defined all the arguments required for the python script.
- the application command corresponding to the entire path of the PyCOMPSs workflow.

The code of the feature extraction workflow is available at:

https://github.com/eflows4hpc/workflow-registry/tree/main/PillarII/feature_extraction_wf/feature_extraction


Save topology.yml

```yaml
tosca_definitions_version: alien_dsl_3_0_0

metadata:
  template_name: TestCmcc1
  template_version: 0.1.0-SNAPSHOT
  template_author: sonia

description: ""

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

topology_template:
  node_templates:
    PyCOMPSJob:
      metadata:
        a4c_edit_x: "-34"
        a4c_edit_y: 3
      type: org.eflows4hpc.pycompss.plugin.nodes.PyCOMPSJob
      properties:
        environment:
          endpoint: "zeus.cmcc.scc"
          user_name: ss18121
        submission_params:
          qos: debug
          python_interpreter: ""
          num_nodes: 1
          extra_compss_opts: "--project_name=$project_name --queue=$queue --streaming=FILES --master_working_dir=$application_path/src --worker_working_dir=$application_path/src
              --worker_in_master_cpus=0 --graph=true --exec_time=$exec_time --env_script=$application_path/src/env.sh --pythonpath=$application_path/src --keep_workingdir"
        application:
          container_opts:
            container_opts: "\"-e\""
          arguments:
            - "$application_path/input/"
            - "$application_path/output/"
          command: "$application_path/src/feature_extraction.py"
        keep_environment: true
      requirements:
        - dependsOnAbstractEnvironmentExec_env:
            type_requirement: environment
            node: AbstractEnvironment
            capability: eflows4hpc.env.capabilities.ExecutionEnvironment
            relationship: tosca.relationships.DependsOn
    AbstractEnvironment:
      metadata:
        a4c_edit_x: 194
        a4c_edit_y: 5
      type: eflows4hpc.env.nodes.AbstractEnvironment
  workflows:
    customWorkflow:
      inputs:
        user_id:
          type: string
          required: true
        vault_id:
          type: string
          required: true
        num_nodes:
          type: integer
          required: true
        project_name:
          type: string
          required: true
        application_path:
          type: string
          required: true
        queue:
          type: string
          required: true
        exec_time:
          type: integer
          required: true
      steps:
        PyCOMPSJob_submit:
          target: PyCOMPSJob
          activities:
            - call_operation: tosca.interfaces.node.lifecycle.Runnable.submit
          on_success:
            - PyCOMPSJob_run
        PyCOMPSJob_run:
          target: PyCOMPSJob
          activities:
            - call_operation: tosca.interfaces.node.lifecycle.Runnable.run
```

Figure 17. TOSCA template for feature extraction workflow.

5.7. Next steps towards the end of the project

There are two main steps towards the end of the project, as stated in the previous deliverable. The first step concerns the finalization of the scientific workflow and its improvement in terms of performance. In particular, the activities are:

- Comparison between the TC detection approaches and the validation of the results.
- Testing of the workflow results with respect to other state-of-the-art methodologies

The second step is about the full integration of the workflow into the eFlows4HPC software stack. In particular, this includes:

- Integration of the Data Logistics Service (DLS) for managing data movement between different computing infrastructures.

- Publication of the workflows on the project workflow registry.

6. Conclusions

This document described the activities performed during the Iteration 2 - Phase 3 of the eFlows4HPC project in order to support the two planned workflows for climate datasets analysis. The activities led to complete workflows in both cases, supported by the eFlows4HPC software stack. Further refinements and final optimizations still need to be made to achieve final releases of fully operational and validated workflows; this aspect will be addressed in the Iteration 2 – Phase 4 of the project.

7. Acronyms and Abbreviations

A4C – Alien4Cloud

ANN - Artificial Neural Network

API – Application Programming Interface

CPU – Central Processing Unit

DLS - Data Logistics Service

ESM – Earth System Model

GPU – Graphic Processing Unit

HPC – High Performance Computing

HPCWaaS – HPC Workflow as a Service

HPDA – High Performance Data Analytics

IBTrACS - International Best Track Archive for Climate Stewardship

ML – Machine Learning

SSH – Secure SHell

TC – Tropical Cyclone

TSMN - Minimum surface temperature

TSMX - Maximum surface temperature

TOC – Table Of Contents

VGG - Visual Geometry Group

VPN – Virtual Private Network

YAML - YAML Ain't Markup Language

8. List of figures and tables

9. References

[1] **Towards HPC and Big Data Analytics Convergence: Design and Experimental Evaluation of a HPDA Framework for eScience at Scale**, in IEEE Access, vol. 9, pp. 73307-73326, 2021, D. Elia, S. Fiore and G. Aloisio, doi: 10.1109/ACCESS.2021.3079139

[2] **PyCOMPSs-Hecuba**: Parallel computational workflows in Python, Enric Tejedor, Yolanda Becerra, Guillem Alomar, Anna Queralt, Rosa M. Badia, Jordi Torres, Toni Cortes, Jesús Labarta, IJHPCA 31(1): 66-82 (2017), DOI: 10.1177/1094342015594678

[3] **COMP Superscalar, an interoperable programming framework**, SoftwareX, Volumes 3–4, December 2015, Pages 32–36, Badia, R. M., J. Conejero, C. Diaz, J. Ejarque, D. Lezzi, F. Lordan, C. Ramon-Cortes, and R. Sirvent, DOI: 10.1016/j.softx.2015.10.004

[4] **Enabling dynamic and intelligent workflows for HPC, data analytics, and AI convergence**, Future Generation Computer Systems, Volume 134, 2022, Pages 414-429, Jorge Ejarque et al., https://doi.org/10.1016/j.future.2022.04.014

[5] **Alien4Cloud**: https://alien4cloud.github.io/

[6] **YORC**: https://github.com/ystia/yorc

[7] **ESM-Tools version 5.0: a modular infrastructure for stand-alone and coupled Earth system modelling (ESM)**, Barbi, D., Wieters, N., Gierz, P., Andrés-Martínez, M., Ural, D., Chegini, F., Khosravi,

S., and Cristini, L.: , Geosci. Model Dev., 14, 4051–4067, https://doi.org/10.5194/gmd-14-4051-2021, 2021.

[8] **Workflow Automation for Cycling Systems**, H. Oliver et al., Computing in Science & Engineering, vol. 21, no. 4, pp. 7-21, 1 July-Aug. 2019, doi: 10.1109/MCSE.2019.2906593

[9] **Cassandra** — A Decentralized Structured Storage System April 2010 ACM SIGOPS Operating Systems Review 44(2):35-40 DOI:10.1145/1773912.1773922

[10] Sidorenko, D., Rackow, T., Jung, T., Semmler, T., Barbi ,D., Danilov, S., Dethloff ,K., Dorn, W., Fieg, K., Goessling, H. F., Handorf, D., Harig, S., Hiller, W., Juricke, S., Losch, M., Schröter, J., Sein, D. V., and Wang, Q., 2015. Towards multi-resolution global climate modeling with ECHAM6–FESOM. Part I: model formulation and mean climate. *Climate Dynamics*, *44*(3-4), pp.757-780.

[11] **TSTORMS**: Detection and Diagnosis of Tropical Storms in High-Resolution Atmospheric Models, https://www.gfdl.noaa.gov/tstorms/

[12] **SLURM**: https://slurm.schedmd.com/documentation.html

[13] **The NetCDF Data Model:**

https://docs.unidata.ucar.edu/netcdf-c/current/netcdf_data_model.html

[14] **TensorFlow**: https://www.tensorflow.org/

[15] **Numpy**: https://numpy.org/

Name of document	***Description***
D5.1	Requirement document that describes a comprehensive list of requirements for the Pillar II use cases, delivered by WP5
D5.2	This document collects the design aspects and technical details of the most relevant use cases considered for the implementation of the Pillar II workflows, delivered by WP5
D5.3	This deliverable relates to the software and documentation released at the end of Phase 1 for the implementation of the Pillar II use cases.