D5.2 Design of the Pillar II use cases

Version 1.0

Documentation Information

Contract Number	9555558
Project Website	www.eFlows4HPC.eu
Contractual Deadline	31.08.2021
Dissemination Level	PU
Nature	R
Author	Julian Rodrigo Berlin (BSC)
Contributors	Suvarchal K. Cheedela (AWI), Alessandro D'Anca (CMCC), Donatello Elia (CMCC), Enrico Scoccimarro (CMCC), Giovanni Aloisio (CMCC)
Reviewer	Nikolay Koldunov (AWI)
Keywords	software design , architecture, data standards, ESM workflows

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	First draft with a proposed ToC
V0.2	First review & improvements
V1.0	Final version, suggestions and corrections from the reviewer were addressed

Table of Contents

1. Executive Summary

This document presents the work done in WP5 concerning the design aspects that should be considered during the implementation of the Pillar 2 relevant use cases, in the context of the eFlows4HPC project. These design decisions concern both the Dynamic ESM (Earth System Model) workflow, the HPDA (High Performance Data Analytics) workflow and Feature extraction functionalities which functional requirements were specified in the D5.1 deliverable.

By design we mean the set of technological definitions that will guide the high level design of the software components that needs to be developed. The design is constrained to the architecture defined for the project described in D1.1 deliverable from WP1 that is briefly described in the first sections of this document for reference purposes.

We divide the design process in two parts; in both cases we list the involved Pillar II use cases and then, for the ESM Dynamic workflow, we describe the workflow by providing a detailed activity diagram at the beginning based on the steps described in the D5.1 deliverable, followed by detailed descriptions and high level implementation details of each of the sub-workflows of the main one, with the idea of guiding the reader through the different design aspects of the ESM workflow steps. In a separate section we describe the design considerations of the Dynamic Analysis part of the ESM workflow where we set the scientific grounds and different aspects that need to be considered towards the development. For the HPDA and feature extraction use cases, the design process consists on describing the three main workflows that were specified in the D5.1 deliverable, by providing detailed descriptions of these workflows, the involved building blocks and its implementation details and also information concerned the CMCC-CM3 datasets that will be used as an input.

As the final step, we provide a mapping of the involved technologies and the software stack components.

This document will serve as a reference during the development phase, as all technical work done in the project will be required to follow these guidelines.

2. Introduction

In this document we make a description of the overall high level design of the Pillar 2 Use Cases as a part of the eFlows4HPC project by providing the following information:

- **The involved use cases and its overall underlying design.** With design we mean the set of technical decisions that will lead to the development of the components that will be integrated in the novel eFlows4HPC software stack, how these will interact with each other, and the scientific background behind these design ideas.
- **Information about integration with the eFlows4HPC software stack** in a high level way, implementation details concerning the novel eFlows4HPC software stack will be provided as a form of insight for the upcoming development phase of the project with the idea to exploit the advantages of the software stack.

- **Information about involved data structures.** Climate models use several types of datasets, the idea is to provide an insight on these formats and the involved storage technologies provided by the novel eFlows4HPC software stack.

2.1 Purpose and scope of the document

This document only applies to the eFlows4HPC Pillar II use cases and requirements, and only includes those Use Cases identified for this working package.

2.2 Reference documents

Name of document	***Description***
D1.1	Architecture document that describes the most important global technical decisions taken for the implementation of eFlows4HPC project, delivered by WP1
D5.1	Requirement document that describes a comprehensive list of requirements, for the Pillar 2 use cases, delivered by the WP5

3. High level view of the architecture

In this section we provide a brief overview of the most relevant architectural decisions taken for the eFlows4HPC software stack described in deliverable D1.1. These decisions will constraint and influence the design of the software components for the Pillar 2 use cases implementation.

3.1 Overview

Figure 1 provides a high level view of the eFlows4HPC software stack modules and most relevant components.

From D1.1 document, a workflow implementation consists of four main parts:

- A description about how the software components are deployed in the infrastructure (provided by an extended TOSCA definition).
- A high level workflow implemented in Yorc (Ystia orchestrator) that executes the deployment and launches different components and also executes the DLS (Data Logistic Service) needed to move the data to be used in the simulation.
- A low level workflow implemented in the PyCOMPSs programming model that will carry out the tasks related to the ESM simulation.
- Data logistics pipelines to describe data movement and transformations to ensure the workflow data is available in the computing infrastructure when required.

All the related workflow artefacts such as software components, datasets and the workflow definitions themselves will be stored in the software stack catalogues and registries (as displayed in Figure 2).

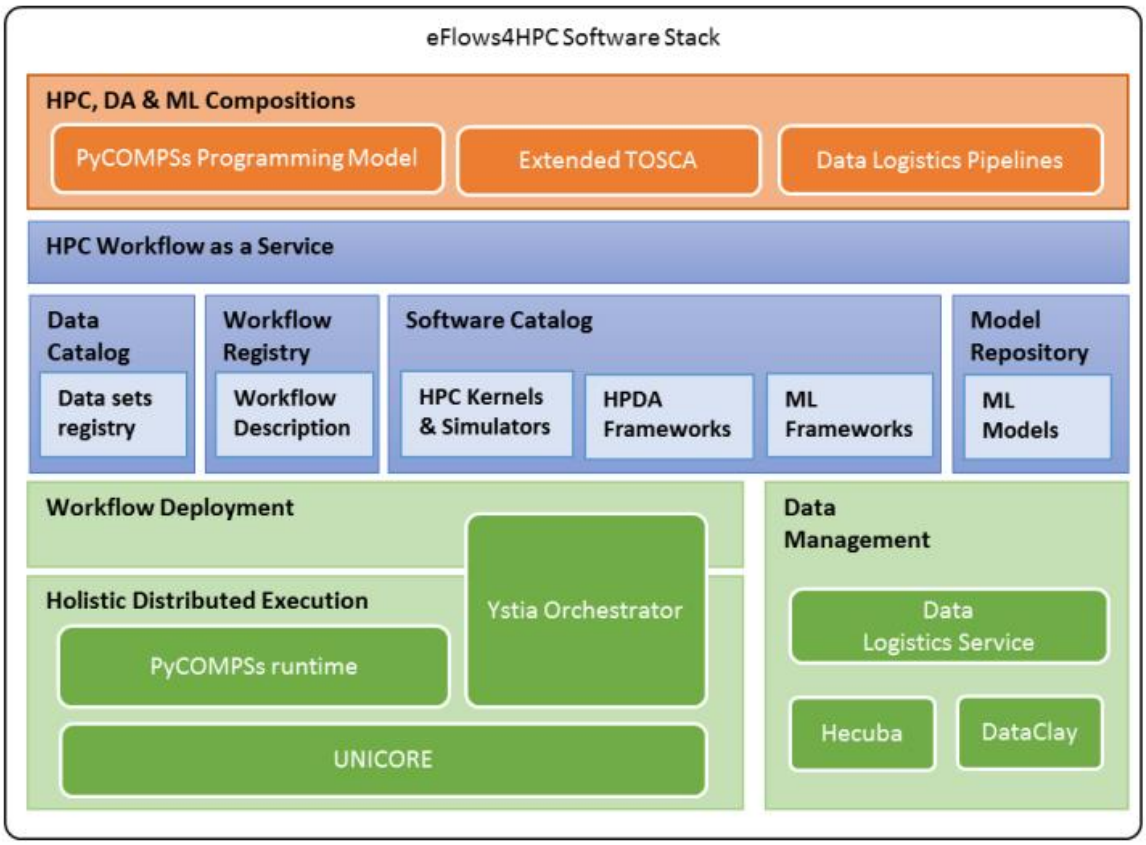

Figure 1 - eFlows4HPC Software Stack Overview

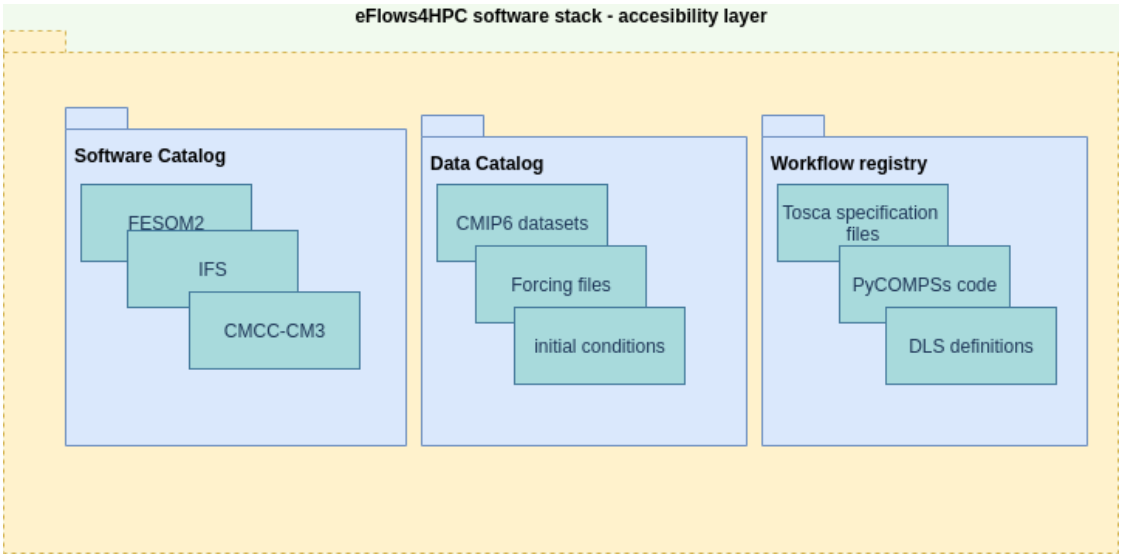

Figure 2 - Diagram showing how the different software components of the Pillar II will be registered in the eFlows4HPC accessibility layer

3.2 Architecture of the Pillar use-cases

In this section we describe the overall design of the software components that are going to be developed in the upcoming phases of the project.

The workflow architecture that enables all the use cases of this WP can be broadly modularized and managed using two applications, a workflow manager and a task manager. Workflow manager is essentially YORC software and task manager is a Python application that heavily uses PyCOMPS to execute its component tasks.

While the development of a workflow manager application and the choice of its components are explored in detail as part of WP1, we supply an envisioned summary of its role to provide a context to this document and give a perspective on interactions among components of the eFlows4HPC software stack.

The workflow manager application coordinates and supervises a use-case workflow. The manager application, essentially YORC software, fetches a use case specific workflow description from a Workflow Registry. A workflow registry may contain many workflows for different use cases.

Each workflow specification, apart from describing component-tasks, their initialization, and interactions, includes reference to a specification of underlying compute topologies that may be used for orchestrating the tasks of a workflow. For instance, ESM simulations might use CPUs, and its diagnostic components may use GPUs. TOSCA specification standard serves as a base-reference to compose such a workflow's topology descriptions involving hybrid infrastructures. The specification has to be extended to accommodate additional ESM-specific needs of the project. Ystia orchestrator (YORC), an envisioned core component of the workflow manager application, uses the workflow specification to initialize necessary compute and storage resources, and orchestrate component tasks of a workflow.

Use cases of this WP can be conveniently launched as a group of tasks by a workflow manager that are coordinated by a task manager application. For instance, in a dynamic ESM workflow, a task manager application coordinates a complex chain of tasks starting from fetching ESM model sources, compilation on target architecture specification, fetching necessary input datasets, generation of ensemble members, their simulation and associated diagnostics. Usage of resources is monitored and any released resources can either be used by task manager or workflow manager to launch new tasks. A task manager is essentially a Python application using PyCOMPSs to compose component tasks.

Fetching input data, sharing data among tasks and publishing value added data of a workflow will be handled using data logistics services.

4. Pillar II Use Cases design

In this section we provide the most relevant design aspects in the form of different diagrams, tables and views in order to show the ideas behind these and other related technical details that we should contemplate towards the development.

As it was described in the deliverable D5.1, the main steps (some of them may be optional) in an order that is close to the chronological for the execution of an ESM workflow are as follows:

- Preparation of model computational mesh

- Preparation of initial conditions
- Preparation of forcing data
- Model compilation
- Preparation of model configuration
- Model run
    - Monitoring of the model run
    - Dynamic data analysis
    - Data output
- Data post-processing
- Data analysis
    - Sanity checks/ standard diagnostics
    - Scientific analysis of the data
    - Feature extraction
- Data archiving
- Data distribution

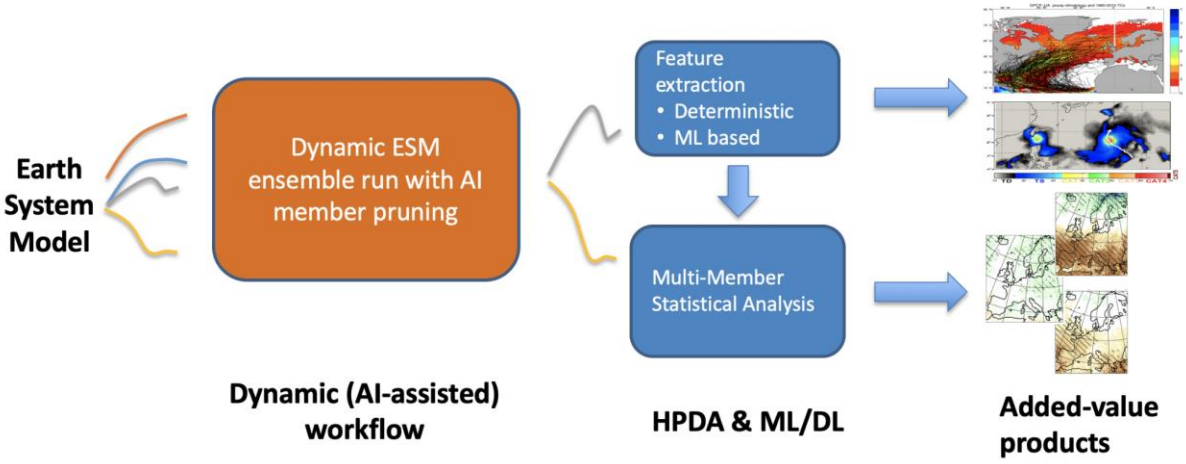

Figure 3 - High level idea of the Pillar II workflows and its three main parts

From Figure 3, there are three main parts to consider:

- The implementation of the ESM Workflow to run the concerned climate models within the software stack;
- The implementation of the Dynamical data analysis that will lead to the member pruning and optimization of the resources using ML techniques;
- The implementation of the HPDA and the Multi-member statistical analysis and feature extraction that takes the output of the workflow as an input.

4.1 Scope of the Design

The intended scope will be the most relevant technical aspects we should consider towards the development. These decisions need to be made in order to implement the mentioned use cases for Pillar 2 taking as a base the defined architecture of eFlows4HPC.

The scope of the design will only cover the use cases mentioned in D5.1

4.2 ESM Dynamic workflow

In this section we provide a summary of the involved use cases for the implementation of the Dynamic ESM Workflow, different diagrams and any other useful illustrations to display different aspects of the design. This includes the first two parts, the ESM workflow and the Dynamical Data Analysis (for the pruning of the members).

The overarching use case of the ESM dynamic workflow component of this WP is to conduct a large number of ensembles simulations that are dynamically pruned, described succinctly in the proposal and D5.1. With this use case as a reference, we develop components of the ESM workflow architecture that are modular and extendable to new use cases.

The target ESM model used for the proposed use cases are OpenIFS/FESOM2, hence this section mostly deals with this ESM.

4.2.1 Summary of involved use cases

The overarching ESM dynamic workflow use case can be broadly split into (sub) use cases, as mentioned in D5.1. They are re-articulated here to provide a context to the following sections.

- Use case 1 (UC1): This use case aims to develop interactions among main ESM workflow components using OpenIFS/FESOM2. Its primary purpose is to design a modular workflow architecture for involved component tasks, namely ESM model setup, ensemble generation, ensemble member simulation and ensemble member pruning.
- Use case 2 (UC2): This use case aims to use complex ESM workflows enabled by UC1 to generate novel scientific applications. This will additionally involve AI based workflow components described in the section 4.2.3.

4.2.2 Use case 1: ESM Dynamic Workflow

The following sections we provide insights about the high level design of the ESM Dynamic workflow and provide some technical details concerning its implementation using the eFlows4HPC software stack.

4.2.2.1 Introduction

As it was described in section 3.2, a workflow in the software stack will consist mainly in two parts:

- High level workflow that will be basically a YORC workflow (described in YAML) where we will orchestrate how the different modules will be deployed, started and undeployed.
- Low level part implemented in PyCOMPSs for the execution of the simulation and exploit the parallelism in the underlying HPC platform.

Taking as input the steps mentioned in section 4, we can define an activity diagram (described in Figure 4) to illustrate the flow control of the different involved steps; some of these steps will be involved in the high level part of the workflow and some others involved in the low level workflow. We will indicate these details in the next sections and, for such purpose, we split the activity diagram (Figure 4) in sub-diagrams where we provide a table with the building blocks we previously defined during the D5.1 deliverable enriched with the matching activities and its implementation details (based on what was mentioned in section 3.2).

We can divide a ESM Workflow in three main parts:

- The initialization

- The execution and monitoring
- The Postprocessing of the generated data and its archiving and distribution

To implement some of the activities mentioned in Figure 4, we will make use of additional tools (see Appendix A - ESM Tools). In the following sections we also describe some of these, and the way they will be integrated.

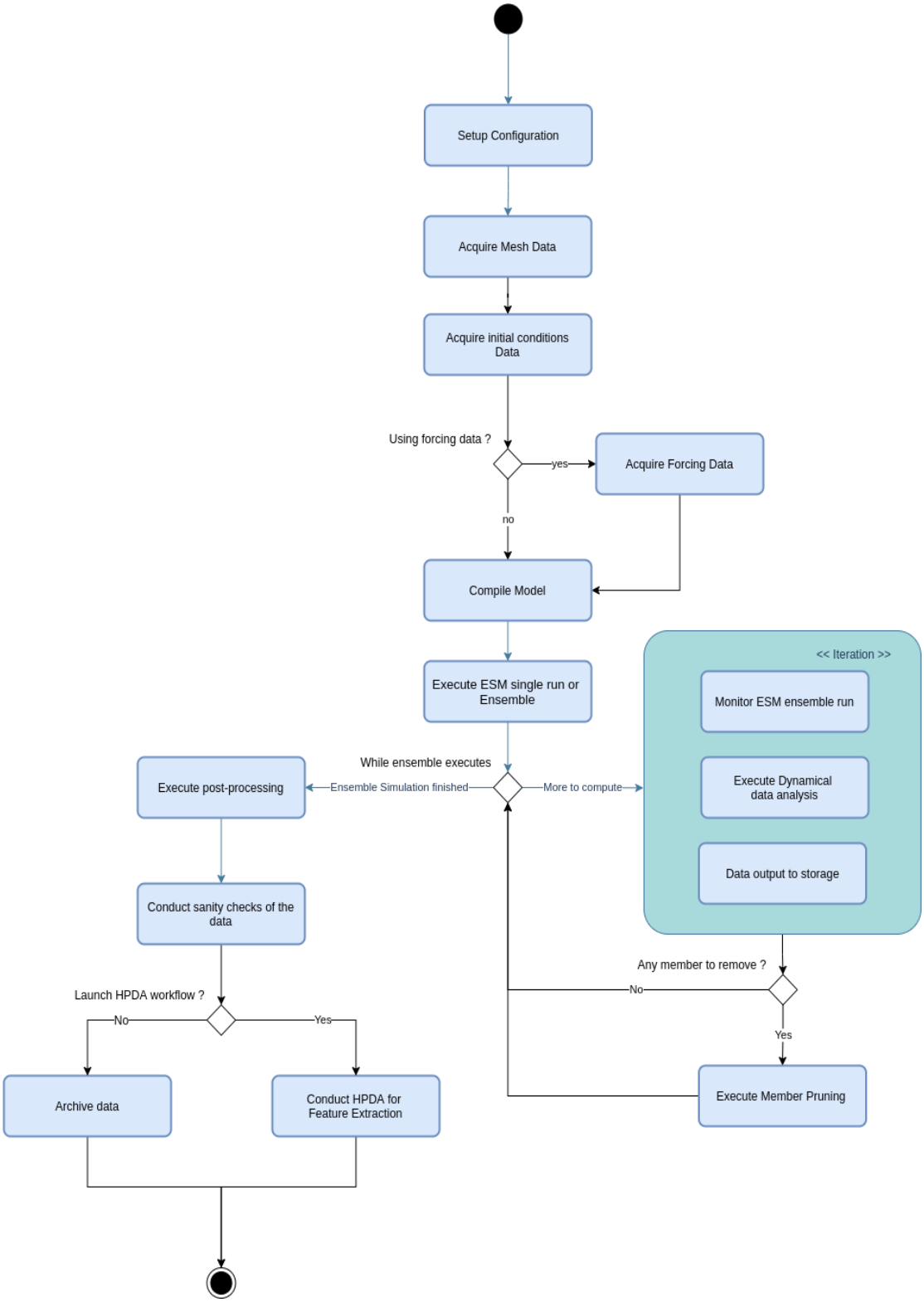

Figure 4 - Activity diagram displaying the overall ESM Dynamic workflow

We will also explore the development of additional features to take advantage of the novel eFlows4HPC software stack that will contribute to the field of earth system modelling, such as strategies of generation of ensembles using non-conventional techniques (described in Section 4.2.2.5) and improvements on the ESM data diagnostics by using data-diagnostic driven tasks and innovative storage solutions (described in Section 4.2.2.6).

4.2.2.2 Workflow Initialization

The following activity diagram (Figure 5) shows the initialization part of the workflow described in Figure 4 each of the defined activities in the diagram maps with one or more building blocks of the ESM workflow, specifically the ones concerned with the initialization.

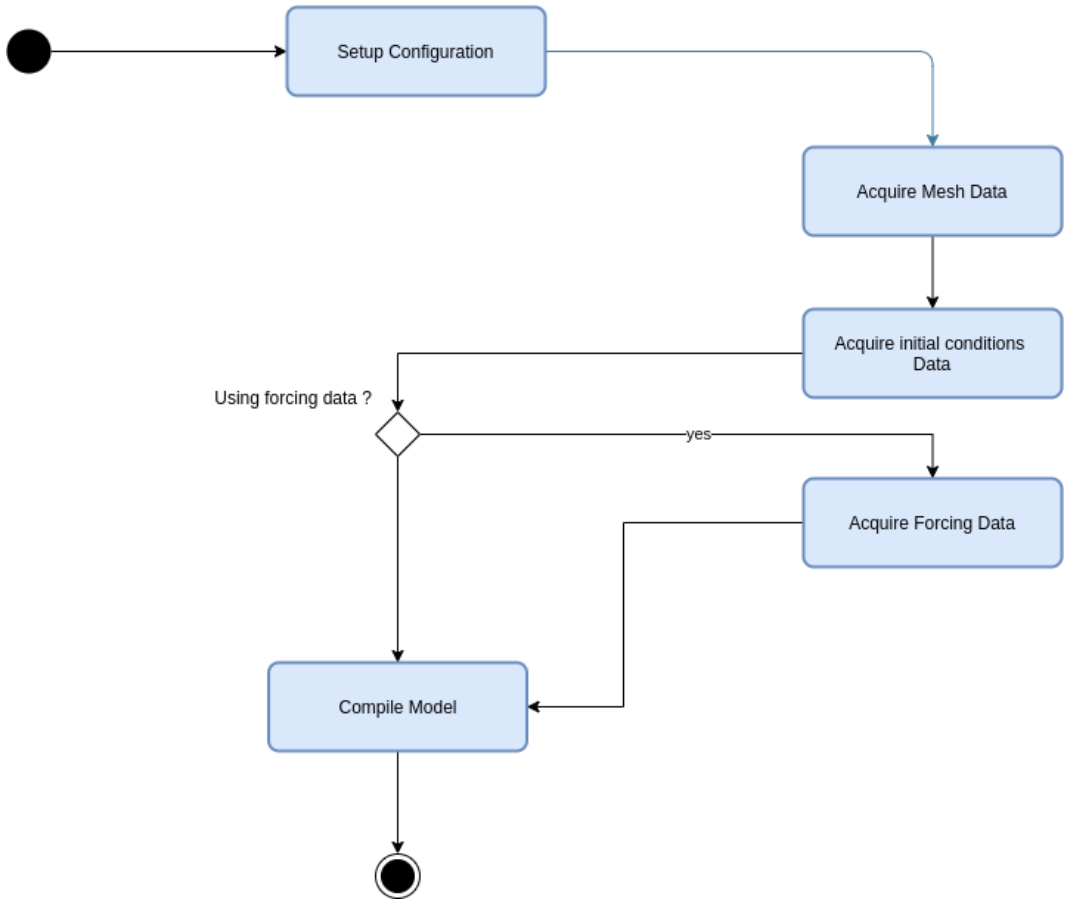

Figure 5 - Diagram corresponding to the initialization part of the ESM workflow prior execution.

For the initialization part, most of the concerned activities will be carried out by the high level part of the ESM-Workflow, that is basically a Yorc orchestration (see sections 3.1 and 3.2), because most of these activities concern the fetch and deployment of software components and the datasets needed to run the model from the software stack repositories.

For some of the tasks to be performed during the initialization, such as the compilation of all the model components on the target platform, we will make use of the esm-tools libraries (described in Appendix A - ESM-Tools) that we will adapt to fit the architecture of eFlows4HPC software stack.

The building blocks concerning the initialization of the ESM-Workflow are displayed in the following table:

Table 1 - Involved building blocks from D5.1 requirement document for initialization

Building Block	Name	Activity	Included actions	Input/Output data structure	Implementation details
1	Deployment of ESM model configuration	**Setup configuration**	Creation and modification of configuration files.	Input: general configuration files. Output: configuration files adjusted for specific model simulation deployed to target location	Once the user has defined the parameters and settings for the ESM ensemble to be run, these files need to be deployed, this will be achieved by executing an activity in Yorc that will transfer the configuration files to the target location. The location of these files may be initially at the local file system of the user or some other repository provided for such purpose.
2	Preparation of model computational grid (mesh)	**Acquire Mesh Data**	Discovery of data location. File copy, grid partitioning if needed.	Input: mesh resolution, mesh type Output: mesh files, ASCII/netCDF, grid partitions.	The meshes or grids are defined in the configuration files for the ESM experiment, for both the ocean and atmosphere components. These components will be retrieved and deployed defining a Data logistic pipeline managed by the DLS (described in Section 3.2), in case the mesh doesn't exist or the partitions need to be generated, it will be done through the proper logic in the previously mentioned pipeline.
3	Preparation of initial conditions	**Acquire initial conditions Data**	Discovery of data location. File copy, sometimes pre-processing to fit the grid.	Input: Climatology or fields from other simulations (ASCII, NetCDF) Outputs: Climatology or fields from other simulations (ASCII, NetCDF)	The initial conditions defined in the configuration for the ESM experiment will be retrieved and deployed through a DLS invocation. We will apply at this step some of the new perturbation strategies such as Randomly perturbed initial conditions (see Section 4.2.2.5) with the goal of generating proper ensembles; these strategies will be implemented as data transformation pipelines. In case that some preprocessing is needed to fit the specified grid, this will be done through the same data transformation pipeline described previously.
4	Preparation of forcing data	**Acquire Forcing Data**	Discovery of data location. File copy, sometimes pre-processing to fit the grid.	Input: Forcing fields of different types (ASCII, NetCDF) Outputs: Forcing fields of different types (ASCII, NetCDF) fitted to the grid.	If the ESM experiment specifies in its configuration files that a given forcing dataset should also be used , then these files will be retrieved and deployed through a DLS invocation, in case that some preprocessing is needed to fit the specified grid, this will be done through a data transformation pipeline within the same invocation.
5	ESM model compilation	**Compile model**	Compilation of all	Input: Model name, Model version,	

			model components, linking libraries.	Output: model executable.	The model sources will be retrieved by the Ystia orchestrator (Yorc) by defining a TOSCA activity that will get the sources and all needed components from the software stack software catalog. Within the same activity will do the compilation and linking of all model sources by using an adapted version of the compilation tools currently presented in the esm-tools (see Appendix A).

Preconditions: All the datasets, model components are correctly configured in the software stack registries and repositories, the configuration for the experiment is already defined and validated

Postconditions: All the needed data is in place and the model already compiled with the specified settings (ESM workflow ready to run)

4.2.2.3 Workflow Execution & Monitoring

In the previous section it was mentioned how the different model components and initialization data will be fetched and deployed prior the execution of the simulation, The following activity diagram (displayed in Figure 6) shows the execution and monitoring part of the workflow described in the Figure 4

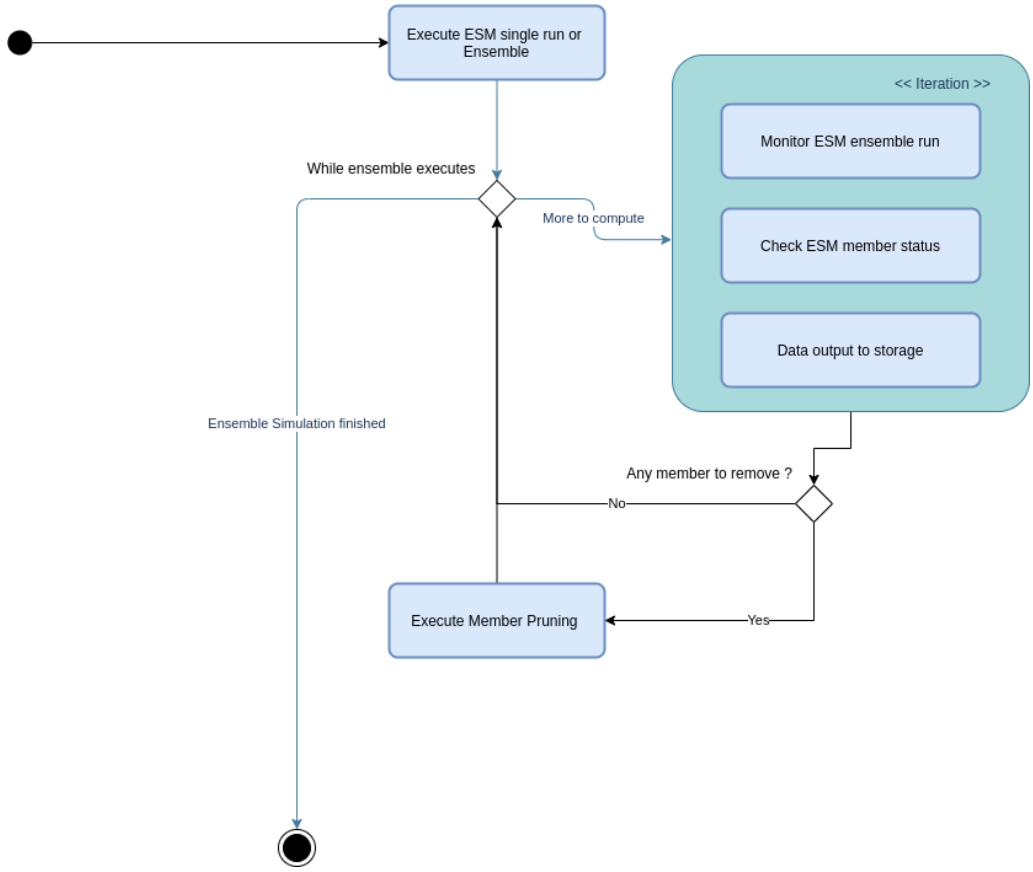

Figure 6 - Diagram corresponding to the execution part of the ESM workflow.

The execution part involves the low level part of the workflow, the task manager provided by the software stack (PyCOMPSs/COMPSs - see Section 3.2) will be in charge to launch the different tasks to execute the simulation and exploit the parallelism in the underlying supercomputing platform. The monitoring part has sides in both, the high and low level workflows, monitoring mechanisms are provided by the software stack, at the high level by Yorc orchestrator and low level by PyCOMPSs.

The results of the dynamic data analysis are accessed periodically during the execution, after a certain number of timesteps have been executed. Such analysis is executed in parallel along with the ensemble and its results stored in Hecuba (see section 4.2.3 for further details).

For running the ESM model we will make use of the esm_runscripts (see Appendix A - ESM-Tools) that we will adapt to fit the underlying architecture and to exploit the new possibilities of the novel eFlows4HPC software stack.

In Table 2 we show the activities defined in Figure 6 and the matching building blocks from D5.1 with some more details concerning the implementation of these activities.

Table 2 - Involved building blocks from D5.1 requirement document for execution & monitoring

Building Block	Name	Activity	Included actions	Input/Output data structure	Implementation details
6	ESM model execution	**Execute ESM single run or Ensemble**	Execution of the model on available resources (see also 5.2 and 5.3 from D5.1).	Input: grid, initial conditions, forcing (netCDF, ASCII) Output: model results Hecuba data snapshot, check points, monitoring results, logs	Model execution will be implemented fully in PyCOMPSs, we can see PyCOMPSs as an additional layer for execution (see Section 3.2 for further information), The esm_runscripts tools (see Appendix A.) will be used to run the simulation. These are Python scripts and will be integrated with PyCOMPSs.
		Check ESM member status	Online data analysis (see 5.2 from D5.1).	Input: Experiment ID Output: array with the members to be discarded	A PyCOMPSs script will be generated to check the state of the running ESM members in Hecuba, this will trigger the pruning and release of the resources.
		Data output to storage	Online data analysis (see 5.2 from D5.1).	Input: raw model data Output: Hecuba data format	Intermediate results will be stored in Hecuba. For such purposes I/O Routines of FESOM2 will be adapted to save the raw model data to Hecuba as part of the implementation of the Data Diagnostics Utils (see Section 4.2.2.6). These changes will also be used to facilitate the dynamic analysis and the pruning process (see Section 4.2.3). During the post-processing phase, this data will be converted to NetCDF format or other convenient standard.

		Execute Member pruning	Online data analysis (see 5.2 from D5.1).	Input: array with the members to be discarded Output: N/A	The pruning of the members will also be carried out fully in PyCOMPSs and also it will be implemented the removal of the data generated by the pruned members and the release of the used resources by these.
		Monitor ESM Ensemble run	Monitoring of the results, and execution. Resubmission of jobs.	Input: Experiment ID Output: Status information of the ongoing simulation	Monitoring mechanisms both for the low and high level workflows will be provided by the software stack.

Preconditions: All model components are deployed and correctly configured, the settings for the experiment were validated and all allocations have been made.

Postconditions: Climate model simulation executes successfully and all the results are ready in the storage.

4.2.2.4 Workflow Post-Processing & disposal

In this section we describe the last part of the workflow (displayed in Figure 7) that concerns the post-processing of the generated data and disposal of the used resources by the simulation, at this stage, we already have the outputs of the simulation in the file system of the HPC (usually located in the scratch of the HPC) or in Hecuba, ready for use.

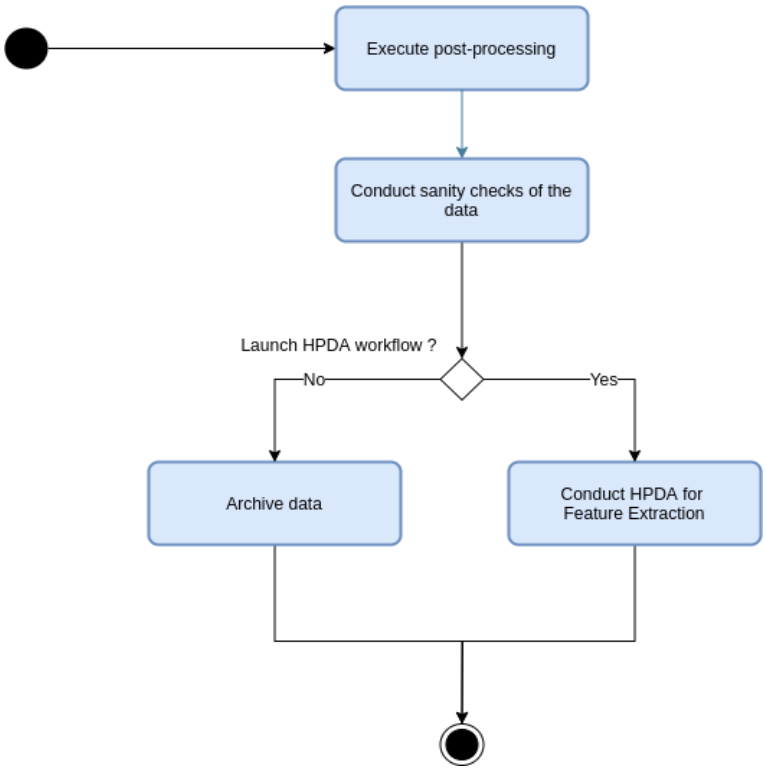

Figure 7 - Diagram corresponding to the post-processing and data handling parts of the ESM workflow

For the post-processing part, most of the logic will be implemented in the low level part of the workflow (PyCOMPSs - see section 3.2 for further details); these scripts will make use of the Data Diagnostics utils (Described in Section 4.2.2.6). After the post-processing and sanitization of the data have took place, it can be either the entry point to other workflows for further scientific analysis such the HPDA and Feature extraction (described in section 4.3) or the data can be just archived and the used resources released (removal of scratch folders used by the simulations, uninstall of all model software components among other things). These tasks will be part of the high level workflow that will deal with data logistics services and the TOSCA activities concerning the undeployment and release of the used resources.

In table 3 we show the activities defined in figure 7 and the matching building blocks from D5.1 (as it was done in the previous sections) with some more details concerning the implementation of these activities.

The workflows concerning HPDA and Feature extraction are fully described in section 4.3 of this document.

Table 3 - Involved building blocks from D5.1 requirement document for the post-processing

Building Block	Name	Activity	Included actions	Input/Output data structure	Implementation details
1	Initial data post-processing	**Execute post-processing**	Conversion of data to different formats for further analysis. Deriving additional variables Adding metadata Data transfer to disc partition where they can be analysed. Initial archiving	Input: model data snapshot in Hecuba format. Output: post processed data transferred to disc partition where analysis can be performed (netCDF, zarr)	The final result is the data fully prepared for further scientific analysis, the post-processing will be implemented in PyCOMPSs taking advantage of having the data stored in Hecuba, logic for performing data conversions (for example from Hecuba format to NetCDF), deriving additional variables and adding metadata in case is needed will be implemented as part of the Data Diagnostics utils development (See Section 4.2.2.5). Any transfer needed for moving the data to a different location before executing the post-processing will also be handled by PyCOMPSs code.
2	Data analysis	**Conduct sanity checks of the data**	initial sanity check Running standard diagnostics Interactive/exploratory DA Scientific analysis (e.g.	Input: Post-processed data (netCDF, zarr) located in the local file system. Output: Results of data analysis as data files netCDF, ASCII or images.	The sanity checks will be implemented the same way, in PyCOMPSs as part of the Data Diagnostics utils (see Section 4.2.2.5). This also can be the entry point (optionally) for the HPDA feature extraction workflow (see section 4.3 for further information).

			Feature extraction, see 5.3 from D5.1)		
3	Data archiving	**Archive data**	Copy data to archive Record information on where data can be found and how they can be retrieved.	Input: Post-processed data (netCDF, zarr) Output: Same data, but located in the target location where these should be stored.	After all the data has been post-processed and sanitized, these should be removed from the HPC file system and archive for later usage and further analysis. At this stage all computations were completed and the results can be transferred via data logistic services defined for such purpose (DLS) in the high level workflow. After all the data is transferred, Yorc will execute the stop and undeployment of the used components and remove these from the HPC. the same of the scratch folders used in the simulations.

Preconditions: at member level, all the computational steps of the simulation have been performed and the generated data by the simulation is already in the storage.

Postconditions: data is post-processed and moved to the target location, and then the involved components are undeployed in the case is the last member of the ensemble being running

4.2.2.5 Ensemble generation utils

A commonly used approach to generate ensembles involves methods used in the research domain of data assimilation such as ensemble Kalman filter [1]. Here we wish to explore ensemble generation strategies that uniquely exploit the architecture of eFlow4HPC and also enable novel use cases.

- **Randomly perturbed initial conditions**: This simple methodology involves applying random perturbations guided by anomalies from a climatology (renalysis/observations) to the input datasets of an ESM. While this is not necessarily the best strategy to generate an ensemble it is easiest to implement and can be used as an initial test case to foster development and integration of workflow components.
- **Perturbed parameter based ensembles**: Parametrizations used in ESM models represent unresolved processes usually involving parameters that are hard to constrain. Perturbing these parameters can be used to generate model ensembles having a desirable, clear association to the physical process. Such ensembles can be used to explore (non-linear) interactions among modelled physical processes and may additionally contribute to an ESM model development (eg., tuning). As such, understanding valid bounds of parameters is of high scientific relevance. Despite these benefits, any reasonable exploration of a state space of parameters of an ESM (several tens) often poses computational challenges. Envisioned workflow architecture of this WP may be used to address this challenge. For instance, starting with a set of parameter perturbations, ML methods may be used on model data to predict best directions to drive the subsequent set of ensembles.

- **Grid configuration based ensembles**: FESOM2, ocean model component of target ESM, OpenIFS/FESOM2 employs an unstructured grid. This allows novel ensemble generation strategies where each ensemble member might have enhanced resolution over different spatial regions while keeping the number of computational grids constant. This might allow effective exploration of parametrization's scale interactions and teleconnections.

4.2.2.6 Data diagnostics utils

ESM data is often subject to mandatory post processing steps before any scientific use. This separation is mainly done to improve computational efficiency of an ESM. For instance, in the case of the atmospheric component model, OpenIFS, used in our target model OpenIFS/FESOM2, it is preferable to save state variables (e.g, vorticity and divergence) in their native spectral space to reduce the number of, compute-intensive, spectral to grid-space transformations during model computations.

Such post processing steps and other ESM data driven pipelines (such as described in section 4.3 and generation of value added products) can be composed as data-diagnostic tasks in an ESM workflow. A conventional and simple approach to enable such a ESM data driven pipeline is to save model output as a file in the filesystem and share it across the diagnostic tasks. But, this often limits the use of heterogeneous computational architectures envisioned in workflows of eFlows4HPC, as each task needs the underlying filesystem to be mounted. We intend to overcome such restrictions by exploring a novel strategy by directly ingesting the data from an ESM into a database that is exposed via a network-accessible API available across all the diagnostic tasks. A proof of concept based on Python, layered on Hecuba, will be developed to explore this aspect.

There has been some initial work in this regard using the OpenIFS model, to ingest data directly in Hecuba. A similar implementation in the FESOM2 model needs to be explored before integrating the entire ESM and benchmarking the performance of this approach.

In summary this data diagnostics component represents a broad framework that facilitates (any) ESM simulation data driven pipelines. Such ESM data driven pipelines are explored in Use case 2, described below.

4.2.3 Use case 2: Dynamical Data Analysis

In this section we describe the initial design and underlying scientific process of the Dynamical Data Analysis that will lead to the optimization of the ESM ensemble in terms of used resources. The functionalities concerned are the ensemble member data analysis and the pruning of the ESM members based on the previous analysis done on the intermediate data of the ensemble.

4.2.3.1 Introduction

Pruning an ensemble member involves composing a pipeline of data diagnostics tasks that lead to a binary metric that is used to make a decision on continuation of the simulation. The metric, at the least, involves determining the validity of the simulated climate state and an estimate of contribution of an ensemble member to (ensemble) prediction skill of the model. A key challenge is to determine the temporal-point of simulation (lead time) to make a decision to prune.

Simplest metric to evaluate the validity of the climate state of an ESM is to compute the global mean, minimum and maximum for a variable such as surface air temperature (a reasonable proxy for evaluating a coupled ESM). This is especially relevant for perturbed parameter based ensembles where there are no guarantees to ensure a valid climate. Similarly, (temporal) signal to

noise ratio (ensemble standard deviation versus ensemble mean of a variable) may be used as an estimate to evaluate contribution of an ensemble member to the (ensemble) prediction skill.

Hindcast experiments allow additional pruning strategies, such as using standard prediction skill scores (e.g, mean square error with respect to reanalysis). These can be used in synergy with perturbed parameter ensembles to determine valid parameter bounds that are of high scientific relevance.

In the next sections we provide more insights about the implementation of the assessment and how this will be calibrated and optimized to achieve good results through scientific experimentation.

4.2.3.2 Ensemble Member data analysis

The analysis of the intermediate ensemble member data will be based on a separate process from the ESM workflow itself that will run in parallel, executing periodically while the simulation progresses. So both will be independent. The logic will be in low level workflow (PyCOMPSs - see section 3.2), and the data produced by the analysis will be consumed by the ESM Workflow in order to execute the pruning of those members that are not useful. In table 4 we can see the associated building blocks defined in D5.1 deliverable together with high level implementation details.

Table 4 - Involved building blocks from D5.1 requirement document for the Dynamic analysis & pruning

Building Block	Name	Included actions	Input/Output data structure	Implementation details
1	Initialization of the ESM Member Diagnostic component	The diagnostic component prepares Hecuba database for the run	Input: Ensemble configuration files, Initial values for the variables to be used (optionally) Output: N/A	The creation of the Hecuba database instance for the simulation will be done at the beginning of the execution of the ensemble as a prior task before executing any simulation task. it will be implemented as a PyCOMPSs task. Config files will be loaded to retrieve useful information to determine some parameters needed for the initialization; these parameters are outlined below in this section.
3	The ESM member diagnostic component conducts the assessment on the data of the ensemble members	Data is retrieved from Hecuba database to conduct the assessment	Input: experiment id, intermediate model data in Hecuba format Output: a list of the members that should be discarded stored in Hecuba	After a certain number of time steps of the simulation have been executed (initial threshold to start launching the assessments is reached), the analysis will be conducted periodically every n time steps, the definition of these parameters is outlined below this section. The logic itself will be implemented in PyCOMPSs in combination with Hecuba, where this information will be stored.

Preconditions: Hecuba and PyCOMPSs runtime are up and running

Postconditions: the database structure is initialized for the ESM ensemble to be run and is ready to be used.

For the initialization part and for the dynamical analysis itself, it is necessary to contemplate the definition of the following variables in the configuration files:

- **ESM_DA_STARTING_POINT**: An initial starting point to conduct the analysis will be also needed. In D5.1 it was pointed out that conducting the assessment in a very early stage of the simulation may lead to inaccurate results.
- **ESM_DA_FREQ**: The Dynamic data analysis (as it was mentioned in the table 4) will execute every n time steps. The determination of frequency by which this analysis will be executed (the number of time steps between each analysis conducted) will be determined experimentally. Setting it too small may lead to an unnecessary overhead and if it is too big the assessment results may be inaccurate and members that may produce useful data may be discarded when they should not.
- **ESM_DA_VARS:** The list of model variables that will be used to conduct the assessment, these will be determined by climate scientists in principle, but we can take those that accumulate value over time as an initial approach, such as *total precipitation (meters of water equivalent per day)* also it is important to limit these to a little number since if we set a huge list it can be very resource consuming.
- **ESM_DA_THRESHOLD**: The value that determines when to discard a given member based on the assessment done. This will also be subject to experimentation to find a proper value.

These variables will be calibrated as a part of the scientific experimental process by running successive ESM simulations with and without the member pruning enabled and these will be compared in order to improve performance of the method.

As it was mentioned in D5.1, the main goal of the assessment is to discard members that will not add anything to the whole ensemble simulation. This can be very useful in huge simulations to release used resources by these members so it can be re-distributed and re-assigned to other processes. There can be two situations concerning the members, either one is too similar to any of the other members (the trend is convergent), or its values are outliers (the trend is divergent - see Table 5 and Figure 8 as example).

4.2.3.3 Similarity criteria

Initially, the idea will be to use accumulated variables as it was pointed out in the previous section (those model variables that aggregate over the time during the length of the simulation), such as *total precipitation* or *Surface Solar Radiation (SSR)* for example. We need to define a similarity criteria in order decide when to mark an executing ESM member for disposal, in order to say how similar or different are two given members in one of its dimensions (a given variable) we can apply a similarity algorithm such as Jaccard similarity [2] or other similar method to calculate how these relate to each other. There are several algorithms for estimating the similarity of a vector of values and several options may be considered since these may impact the performance of the simulation as well so it may be needed to explore different alternatives to find an equilibrium of performance and effectiveness on removing non useful members.

If after a certain number of timesteps, the assessment results on a member whose similarity coefficient compared to the others is bigger than the defined threshold, it will be marked for disposal.

The initial assessment will be based, as mentioned before, on accumulated variables. But the idea is to create a more sophisticated mechanism to use other type of variables and also machine learning techniques to do a more optimal assessment, at this stage it is to early to describe a solution based on machine learning, but it will be envisioned on the upcoming development

phases of the project with the goal to exploit at maximum the capabilities provided by the different techniques and the functionalities provided by the eFlows4HPC software stack.

Table 5 - sample values through the timesteps of a simulation for an accumulated variable with a divergent tendency

Members	Time Steps (days)						
	1	10	20	30	40	50	60
fc0							
Total precipitation (m of water equivalent per day)	10	22	34	37	39	41	45
fc1							
Total precipitation (m of water equivalent per day)	9	30	44	45	67	80	101
fc2							
Total precipitation (m of water equivalent per day)	11	21	30	32	37	46	48
fc3							
Total precipitation (m of water equivalent per day)	12	23,4	27	36	40	48	50
fc4							
Total precipitation (m of water equivalent per day)	14	25	28	32	42	43	45

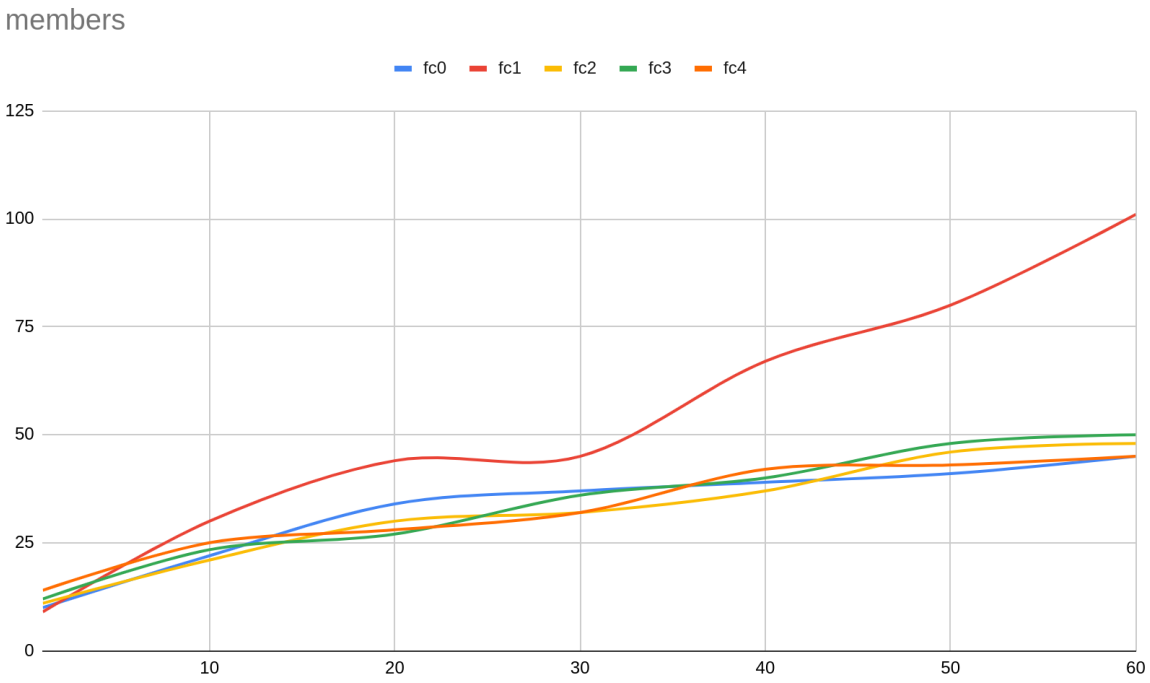

Figure 8 - Divergent member example for a ESM simulation of 5 members per date to show divergence

4.2.4 Data model

FESOM2, the ocean model component of the target coupled ESM model, OpenIFS/FESOM2, uses an unstructured triangular grid configuration, unlike most contemporary ESM models. Its grid configuration, best shown in FESOM2 model documentation [3], differs for scalar and vector variables. As an example, data structure representing model output for a scalar variable is shown below as header of common data format (equivalently NetCDF).

dimensions:

nelem = 243899 ;

three = 3 ;

nod2 = 126858 ;

nz = 48 ;

nz1 = 47 ;

time = 10 ;

variables:

uint faces(nelem, three) ;

double lat(nod2) ;

lat:_FillValue = NaN ;

lat:long_name = "latitude" ;

lat:units = "degrees_north" ;

double lon(nod2) ;

```
        lon:_FillValue = NaN ;
        lon:long_name = "longitude" ;
        lon:units = "degrees_east" ;
    double nz(nz) ;
        nz:_FillValue = NaN ;
    double nz1(nz1) ;
        nz1:_FillValue = NaN ;
    int64 time(time) ;
        time:axis = "T" ;
        time:long_name = "time" ;
        time:standard_name = "time" ;
        time:stored_direction = "increasing" ;
        time:units = "days since 1948-12-30T23:15:00" ;
        time:calendar = "proleptic_gregorian" ;
    float temp(time, nod2, nz1) ;
        temp:_FillValue = NaNf ;
        temp:description = "temperature" ;
        temp:long_name = "temperature" ;
        temp:units = "C" ;
        temp:coordinates = "lat lon" ;
```

While such a grid structure provides novel opportunities (e.g, ensembles based on refining mesh over different regions described in Section 4.2.2.5), it poses additional challenges for computation of data diagnostics. This is because, the most commonly used post-processing tools, as yet, do not allow easy processing and visualization of data on an unstructured grid.

Pyfesom2, a software package used to analyse FESOM2 model output, aims to address this challenge by providing an interface to commonly used tools (in Python) and by providing commonly used diagnostic and visualization methods on an FESOM's unstructured grid. To allow envisioned data diagnostics of this WP, it is necessary to contribute to the software development of Pyfesom2.

4.3 Statistical analysis and feature extraction

This section describes the design of the statistical analysis and feature extraction part of the Pillar II workflow and the involved components and data structures.

4.3.1 Summary of involved use cases

The following sections describe the three main workflows supported by the feature extraction (TCs - Tropical Cyclones) use case of Pillar II. A more detailed description of the various building blocks, the interaction among the components and the structure of the involved data is provided in the next section.

As already mentioned in section 3.2, the entire workflow deployment, initialization and management will be performed by the upper layer of the eFlows4HPC software stack managed by YORC and PyCOMPs; specifically YORC will take care of the deployment and initialization of the workflow while PyCOMPS of the orchestration and execution of the different subtasks.

More in detail, the first use case represents the base scenario where the analysis of TCs tracks is applied on the output of a single model (in this case the CMCC-CM3 ESM), the second one represents a more integrated approach where the analysis of TCs tracks is performed jointly with the HPC model execution (again the CMCC-CM3 ESM), while the third use case focus on the multi-model analysis of TCs tracks in the context of climate data from the CMIP6 experiment. The Coupled Model Intercomparison Project phase 6 (CMIP6) experiment will deliver to the scientific community more than 20PBs of data from climate simulations, which means around 10 times more than the previous phase (CMIP5). The CMIP6 archive provides access to multiple variables with different time and spatial resolutions. In the context of the TC detection analysis, we are interested in the data with the highest resolution (6-hourly data at ¼ degree).

Tropical cyclone detection and tracking can be done by following different tracking methods available in literature (i.e., deterministic approaches) and also by investigating new Machine Learning approaches, to verify the possibility to speed-up the detection process in the contest of a multi-model/multi-member analysis. The following use cases will take into account and compare different approaches with respect to metrics such as accuracy, performance, etc.

4.3.1.1 Use case 1: CMCC-CM3 datasets

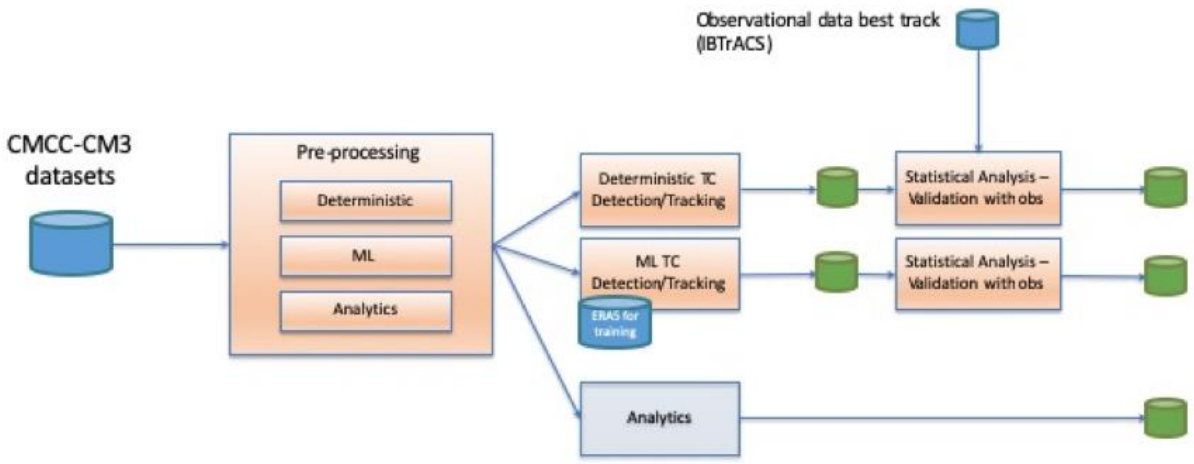

Figure 9 - Use case 1 based on CMCC-CM3 datasets

This first workflow consists of the execution of the feature extraction (Tropical Cyclone detection and tracking procedures and extreme events analysis) on the CMCC-CM3 datasets, already available on storage as input of the subsequent steps. The following phases are then executed:

- Pre-processing: includes a set of preliminary steps to organize/modify/regrid the data accordingly for the following substeps;
- TC Detection and tracking: consists of different approaches (e.g., deterministic and data-driven) for TC detection and tracking. This block produces as output additional higher-level products that can complement the model output;
- Statistical analysis and validation: this step will perform a set of analysis on the output produced by TCs detection/tracking stage, as well as comparison with observational data in order to validate performance (in terms of accuracy) of the various approaches. The output of this step will consist of additional features extracted from the TCs detection/tracking phase.

It is worth mentioning that the ML TC Detection/Tracking block consists of two different stages: a training phase executed offline (outside the workflow execution), where the model is fitted based on reanalysis data (e.g., ERA5), and an inference stage, which is part of the workflow and is executed on the pre-processed CMCC-CM3 data.

The generality of the workflow structure could also allow an easy integration of additional analysis blocks for feature extraction from the model output data (third branch), for example concerning extreme events analysis.

4.3.1.2 Use case 2: CMCC-CM3 runtime

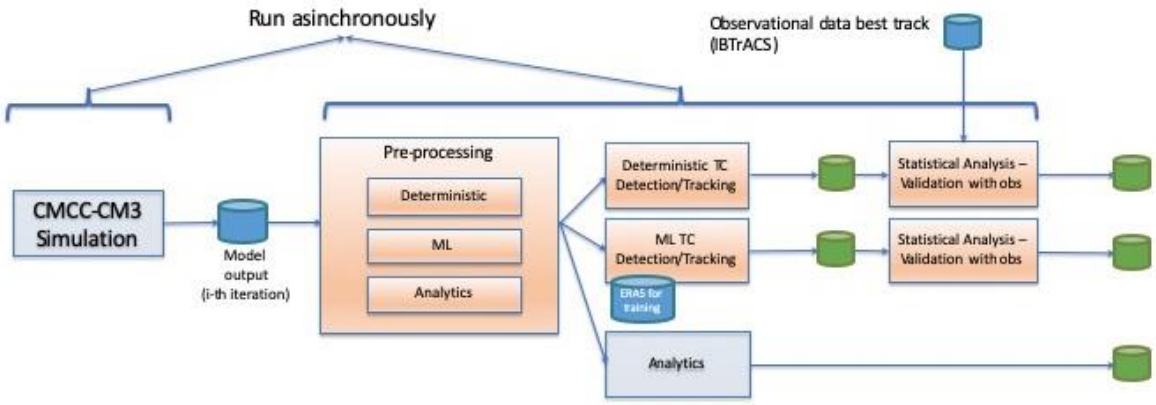

Figure 10 - Use case 2 based on CMCC-CM3 datasets produced at runtime

The second workflow consists of a more integrated scenario where the analysis pipeline is executed on the data produced by the model at running time; the single blocks of the workflow are similar to those defined in use case 1. The pre-processing block in this case gathers the required input data during the model execution. The model simulation and the feature extraction phase run asynchronously in order to incrementally produce the feature extraction results along with the model data

4.3.1.3 Use case 3: CMIP6 datasets

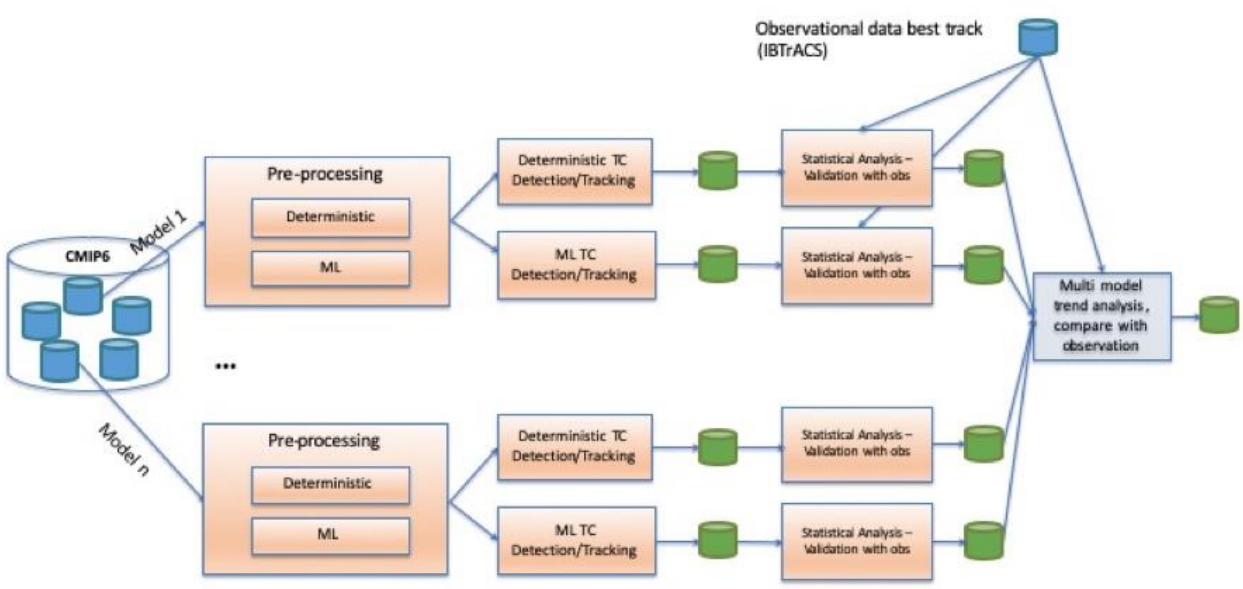

Figure 11 - Use case 3 based on CMIP6 datasets

Similarly to the first use case, this workflow starts from the CMIP6 multi-model experiment data already available on the storage. The workflow consists of multiple independent branches executed on different models (e.g., HighResMip) and experiments (historical, future) from the CMIP6 dataset.

The branches can be executed independently and concurrently. Each branch performs the same steps reported in use cases 1.

A final additional step is then performed on the whole set of intermediate results produced from the various models in a multi-model (ensemble) analysis.

4.3.2 Workflow building blocks description

This section provides a detailed description of the various building blocks involved in the workflows presented in the previous section.

In particular, the following blocks will be described in terms of characteristics, input data, output data and computational requirements:

- CMCC-CM3 model run;
- the different pre-processing phases (able to prepare the datasets for the subsequent analysis);
- the Tropical Cyclones detection modules, based on a Machine Learning and on a deterministic approach;
- the analytics modules, respectively the Extreme Events analysis and the multi-member/statistical analysis phases.

Specifically, concerning the Tropical Cyclones detection exploiting a ML approach, a preliminary training phase is needed in order to set up the proper Neural Network and the related

hyperparameters. This phase is not included into the workflow execution at run time and it is reported as a separated section (paragraph 4.3.2.1).

4.3.2.1 Machine Learning TC detection training

As previously mentioned, the pre-processing stage implements the operations to prepare the ESM data for the following steps.

The ML-based TC analysis relies on two different phases that are *Training* and *Feature Extraction (Inference)*.

- *Training phase*: the TC detection model is implemented, trained and validated on historical TCs data (ERA5 and IBTrACS).
- *Feature Extraction (Inference phase)*: the final trained model is used, into the workflow, to detect TCs on the output of CMCC-CM3 and CMIP6 models.

The training is decoupled from the workflow and represents a prerequisite, while the feature extraction (inference) phase is reported in the next subsections.

The training phase aims at developing a Convolutional Neural Network (CNN) architecture that is able to learn to detect spatial-invariant TCs patterns from historical tracks exploiting the selected climatic fields (as specified in the pre-processing phase). This phase is totally decoupled from the workflow and performed only once. The design of the CNN is ongoing.

Two sub-stages can be identified: preparation of the data and the Neural Network model training.

Pre-processing for the training phase

INPUT DATA

- The **IBTrACS (International Best Track Archive for Climate Stewardship)** dataset provides best track metadata on historical worldwide TCs extreme events
    - Spatial coverage: Global
    - Temporal coverage: 1842 to present
    - Selected geographical domain: 0–70 °N, 100–320 °E
    - Selected temporal domain: 1979 to present
    - Temporal resolution: a record every 3 hours
    - Extracted fields: SID (Storm ID), NAME, ISO_TIME, BASIN, LAT, LON

  For each selected IBTrACS record, the following climatic maps have been downloaded (one map every 3-hours):

    - **ERA5 hourly data on single levels from 1979 to present**
        - Spatial coverage: Global
        - Temporal coverage: 1979 to present
        - Selected geographical domain: 0–70 °N, 100–320 °E
        - Selected temporal domain: 1979 to present
        - Horizontal resolution: 0.25° x 0.25° (~25 km x ~25 km)
        - Temporal resolution: 1 hour

- Extracted fields: 10m wind gust since previous post-processing, Instantaneous 10m wind gust, Mean Sea Level Pressure, Sea Surface Temperature

Notes:

- ➢ "10m wind gust since previous post-processing" climatic field provides, for each grid point, information about the 10m wind gust peak in the last hour. This field has been processed to extract the maximum 10m wind gust within the past 6 hours
- ➢ There is a 1:1 correspondence between the downloaded ERA5 maps and each record of the IBTrACS dataset

- **ERA5 hourly data on pressure levels from 1979 to present**
  - Spatial coverage: Global
  - Temporal coverage: 1979 to present
  - Selected geographical domain: 0–70 °N, 100–320 °E
  - Selected temporal domain: 1979 to present
  - Horizontal resolution: 0.25° x 0.25° (~25 km x ~25 km)
  - Temporal resolution: 1 hour
  - Extracted fields: (Relative) Vorticity at 850 hPa, Temperature at 300 hPa, Temperature at 500 hPa

In the selected geographical domain, 2281 TCs have been identified from the IBTrACS dataset in the 1979–2020 period, for a total of 141375 IBTrACS records that correspond to 3-hours resolution tracks. Then, 66527 ERA5 climatic maps related to the aforementioned climatic variables have been gathered from the Copernicus CDS (Climate Data Store). As an example, a map of Mean Sea Level Pressure (14/9/2019 h. 18.00) has been gathered from the ERA5 dataset according to the information provided in IBTrACS. As shown in the map, the IBTrACS metadata evidenced the presence of two TCs in the considered domain that have been highlighted through dashed red boxes.

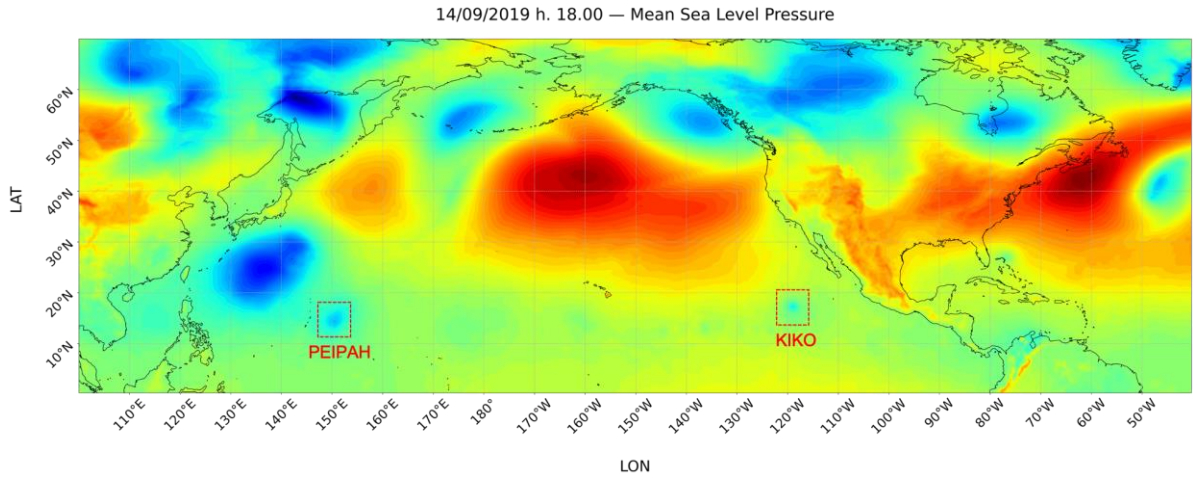

Figure 12 - Map related to a TC detection (14/9/2019 h. 18.00)

OUTPUT DATA

- A NetCDF file for each 3-hourly IBTrACS TC record that integrates the extracted climatic fields from "ERA5 hourly data on single levels from 1979 to present" and "ERA5 hourly data on pressure levels from 1979 to present" for the selected spatial domain
- Additionally, for each TC occurrence in every NetCDF file, a total of 4 patches of size 40 x 40 points each, are generated according to the following procedures:
    - DYNAMIC PROCEDURE: for each TC, it crops from the climatic map one patch containing the TC nearly in its center and a second patch that does not contain the TC
    - STATIC PROCEDURE: for each TC, it crops from the climatic map one patch containing the TC in a random position within it and a second patch that does not contain the TC

  Notes:

  ➢ In order to retain the correspondence between the TC center Latitude/Longitude coordinates and its position within the patch, a georeferencing mapping process has been applied. Specifically, latitude and longitude coordinates have been discretized on the 0.25° x 0.25° ERA5 grid and then converted to the patches coordinates reference system (pixels)

  ➢ These procedures act as data augmentation techniques that can be helpful in the training process of the NN for improving the accuracy and to avoid overfitting

COMPUTATIONAL REQUIREMENTS

- No particular computation requirements are needed
- Storage required:
    - ≥ 240 GB (ERA5 Maps, NetCDF format)
    - ≥ 20 GB (ERA5 patches, NetCDF format)

Training phase

INPUT DATA: IBTrACS records of historical TCs tracks in terms of Latitude and Longitude geographical coordinates of the TC center (output variable/target variable)

Patches of 40 x 40 points each, corresponding to the extracted ERA5 climatic fields (input variables/features)

OUTPUT DATA: Trained model (model and weights saved in HDF or other formats to allow portability)

COMPUTATIONAL REQUIREMENTS:

Hardware requirements: The Training phase requires 1 compute node equipped with 4 x GPU.

Software requirements: The Training procedure will be carried out using Python v3 based on the Keras API and relying on TensorFlow backend. Additionally, the use of the EDDL framework will also be investigated.

4.3.2.2 CMCC-CM3 simulation run

CMCC-CM3 is the latest model version under development at CMCC, based on the previous version of the CMCC-CM2 coupled climate model [7,8], largely based on the Community Earth System Model (CESM) project (http://www.cesm.ucar.edu) operated at the National Centre for Atmospheric Research (NCAR) in the United States, and used to run CMIP6 simulations following both simulation scenarios and HighResMIP protocols. The important and strategic difference with the NCAR coupled model is the oceanic component, which is based on Nucleus for European Modelling of the Ocean (NEMO) model. In CMCC-CM3, the atmospheric component is the CAM6 and the ocean component is NEMO 4.0. The adopted spatial resolution is ¼ degree, corresponding to about 25 km grid spacing.

INPUT DATA: netcdf data representative of the radiative forcing gas concentration are needed, together with initial conditions for the atmosphere, ocean and ice model components.

OUTPUT DATA: model output is collected as monthly files. In the atmospheric component different files are created for each month, corresponding to different time frequency output (from 6-hourly to daily and monthly), containing different multidimensional (latitude x longitude x vertical level x time) fields. For the ocean component the maximum time frequency saved is the daily one.

COMPUTATIONAL REQUIREMENTS: about 1000 CPUs are required to run one model year in two day real time on CMCC Zeus supercomputer. Also about 165GB of model output is generated for each year of simulation.

4.3.2.3 Pre-processing for Machine Learning (inference) TC detection

This block of the workflows takes care of preparing the data for the Machine Learning-based model (inference phase) for Tropical Cyclone detection.

INPUT DATA

- **CMCC-CM3 data (Use cases 1 and 2)**

Table 6 - Correspondence between **CMCC-CM3** and **ERA5** climatic fields

CMCC-CM3	ERA5
WSPDSRFMX	10m wind gust since previous post-processing (fg10) (*)
(**)	Instantaneous 10m wind gust (i10fg)
PSL	Mean Sea Level Pressure (msl)
(***)	(Relative) Vorticity at 850 hPa (vo)
T300	Temperature at 300 hPa (t)
T500	Temperature at 500 hPa (t)
TS	Sea Surface Temperature (sst) (*)

* This variable can be omitted from the analysis if necessary, since it is not a standard predictor for the TCs detection task.
** This field needs to be stored during CMCC-CM3 execution.
*** Computed in post-processing.

- **CMIP6 data (Use case 3)**

Table 7 - Correspondence between CMIP6 and ERA5 climatic fields

CMIP6	ERA5
–	10m wind gust since previous post-processing (fg10) (*)
wind speed (sfcWind)	Instantaneous 10m wind gust (i10fg)
air pressure at mean sea level (psl)**	Mean Sea Level Pressure (msl)
atmosphere relative vorticity (rv850)	(Relative) Vorticity at 850 hPa (vo)
air temperature (ta)	Temperature at 300 hPa (t)
air temperature (ta)	Temperature at 500 hPa (t)
–	Sea Surface Temperature (sst) (*)

* This variable can be omitted from the analysis if necessary, since it is not a standard predictor for the TCs detection task.
** This variable (psl) can assume different nomenclature: air pressure at mean sea level, air pressure at sea level, sea level pressure, sea level pressure.

OUTPUT DATA

The pre-processing procedure of the Feature Extraction consists of several steps to be performed on CMCC-CM3 and CMIP6 data, that comes in NetCDF format:

- Identification of the same set of variables gathered from ERA5 (see Table 1 and 2)
- Spatial consistency preservation with respect to the training data
    - Interpolation of data to achieve the same spatial resolution (0.25° x 0.25°) of training data (e.g. re-mapping operations by means of cdo)
    - The resulting interpolated map is made up of 721 x 1440 grid points (lat, lon). The last latitude is dropped in order to facilitate the process of patch generation
- Generation of patches
    - The interpolated climatic maps are split into 18 x 36 non-overlapping patches of 40 x 40 grid points each that cover the whole input map. This makes the model able to

process each patch in an efficient way and to detect potential multiple TC occurrences in a single time instant, on the same input map

COMPUTATIONAL REQUIREMENTS

- No particular computation requirements are needed
- Storage required:
  - CMCC-CM3: it depends on data stored on disk for the Use case 1 and on the output data of the model simulation for the Use case 2
  - CMIP6: it depends on data stored on disk for the Use case 3 and on the number of models involved in the comparison

4.3.2.4 Pre-processing for deterministic TC analysis

This stage includes the preparation of the NetCDF data for the deterministic TC analysis block.

Field listed in table 6 must be extracted from the CMCC-CM3 model output, or collected from ESGF for the CMIP6 models. 6-hour variables, such as the one listed in table 6, must be concatenated in time to be digested by the tracking deterministic algorithm.

In particular, an example of the header of CMIP6 NetCDF data for the CMCC-CM2-VHR4 model, air temperature variable, is shown below. The file is a portion of the whole dataset limited to 1 month of data (4 time steps per day). The full dataset consists of 780 of such files. As it can be seen the main variable “air temperature” is a 4-dimensional matrix with time, pressure, latitude and longitude.

dimensions:

time = UNLIMITED ; // (124 currently)

plev = 7 ;

lat = 768 ;

lon = 1152 ;

bnds = 2 ;

variables:

float ta(time, plev, lat, lon) ;

ta:standard_name = "air_temperature" ;

ta:long_name = "Air Temperature" ;

ta:comment = "Air Temperature" ;

ta:units = "K" ;

ta:cell_methods = "area: mean time: point" ;

ta:cell_measures = "area: areacella" ;

ta:missing_value = 1.e+20f ;

ta:_FillValue = 1.e+20f ;

INPUT DATA: CMCC-CM3 output data for use case 1 and 2, and CMIP6 datasets for use case 3. Both in NetCDF data format. The necessary fields are the ones indicated in table 6.

OUTPUT DATA: The output consists in the position, in terms of Latitude and Longitude geographical coordinates, of the TC center within each patch and the associated wind speed, covering the global domain.

COMPUTATIONAL REQUIREMENTS: Large amount of main memory, also distributed, to enable fast in-memory processing.

4.3.2.5 Pre-processing for extreme events data analytics

Concerning the data analytics block, the related pre-processing phase will perform a set of operations in order to prepare the CMCC-CM3 or CMIP6 datasets for statistics and indices computation; specifically, a set of indices will be taken into account in the analytics block ('Feature extraction phase', 'Extreme events analytics' section) which require specific input data for the proper and fast computation. Some examples are variable selection, domain subsetting, file/variable concatenation, data structure transformation.

An offline phase will also be performed for extracting long-term (in a range of 30 years e.g., 1961-1990) statistical values such as averages and percentiles for the indices computation and comparison. This phase is not part of the running workflow.

INPUT DATA: CMCC-CM3 output data for use case 1 and 2, and CMIP6 datasets for use case 3. Both in NetCDF data format. Temperature and precipitation variables will be mainly considered for the extreme event analysis.

OUTPUT DATA: Multi-dimensional datasets ready for extreme events data analytics.

COMPUTATIONAL REQUIREMENTS: Large amount of main memory, also distributed, to enable fast in-memory processing.

4.3.2.6 Machine Learning TC detection inference

The Feature Extraction is the Inference carried out within the workflow. It runs the analysis over the pre-processed data. In this phase, the pre-trained Neural Network (Training phase) is exploited to detect and localize TCs center, in terms of Latitude and Longitude geographical coordinates, on CMCC-CM3 and CMIP6 data, as described by Use cases 1, 2 and 3 respectively.

INPUT DATA

- For Use cases 1 and 2, CMCC-CM3 data divided into non-overlapping patches of size 40 x 40 points each (re-mapped at the resolution of 0.25° x 0.25°) as described in the pre-processing Section
- For use case 3, CMIP6 data divided into non-overlapping patches of size 40 x 40 points each (re-mapped at the resolution of 0.25° x 0.25°) as described in the pre-processing Section

OUTPUT DATA

- For all the Use cases the output consists in the position, in terms of Latitude and Longitude geographical coordinates, of the TC center within each patch. The output file will be delivered in NetCDF and CSV formats

COMPUTATIONAL REQUIREMENTS

- The Feature Extraction requires 1 compute node equipped with 4 x GPU

4.3.2.7 Deterministic approach for TC detection

The TC occurrence method used is a tracking technique looking for individual TCs based on objective criteria for the identification of specific atmospheric conditions based on [4]. In particular, a two-step procedure is applied:

1) Potential storms are identified based on the three following criteria:

(i) For each 6-h time step, the grid points where the relative vorticity at 850 hPa exceeds the threshold of 1.6 3 1024 s21 are identified.

(ii) If a local sea level pressure minimum is located within a distance of 28 latitude or longitude from the vorticity maximum defined in the previous criterion, the relative grid point is considered as the center of the storm. In addition, the local maximum of the 10-m wind speed within the 6-h step is recorded.

(iii) The TC warm core is defined based on the temperature averaged between 300 and 500 hPa. Only a warm-core temperature greater than 18C with respect to the surrounding mean temperature (over a 68 latitude 3 68 longitude box) is considered as associated with aTC condition. The distance of the warm-core center from the storm center must be within 2 degrees.

2) Storms are tracked as follows: for each potential storm condition, the algorithm verifies the presence of storms during the following 6-h time period within a distance of 400 km. If no storm is found, the trajectory is considered finished. If any storm is detected, the closest storm is chosen as belonging to the same trajectory as the initial storm. To qualify a tracked trajectory as a storm, it must last at least 3 days and have a maximum surface wind speed greater than 17ms-1 during at least 3 days, without any constraint regarding their timing during the TC evolution. This tracking algorithm has been validated as capable of realistically representing TC activity in previous studies [5][6][7].

INPUT DATA: CMCC-CM3 model output or CMIP6, 6-hourly model output concatenated in time after the selection of the needed fields (see Table 6).

OUTPUT DATA: The output consists in the position, in terms of Latitude and Longitude geographical coordinates, and the associated wind, of the TC center at the 6h frequency, for all the detected TCs . The output file will be delivered in NetCDF format.

COMPUTATIONAL REQUIREMENTS: There is no need for parallelization since the code is sufficiently fast to be run on a single CPU. The main requirement is the storage for input data (order of magnitude of ten GB for each year of simulation analysed).

4.3.2.8 Extreme events analytics

This stage will perform a set of statistical and mathematical operations to compute extreme events indices starting from some examples from the ETCCDI Climate Change Indices lists[1] (e.g., heat wave duration/frequency/magnitude etc.). For example, heat waves can be defined as a period of three consecutive days where the maximum temperature is above a reference daily threshold [8]. To this end, different types of time-series oriented operations will be performed over multiple variables. Long-term statistics will also be involved in the computation as reference/threshold values for some of the metrics (e.g., long-term averages or long-term percentiles). This type of analysis can benefit from parallel processing since the same computation can be applied independently on each time series of the multi-dimensional data.

[1] http://etccdi.pacificclimate.org/docs/ETCCDMIndicesComparison1.pdf

INPUT DATA: Multi-dimensional datasets with temperature and precipitation variables ready for extreme events data analytics, as well as the long term statistical information from the pre-processing stage.

OUTPUT DATA: NetCDF data and maps with the results of the analysis.

COMPUTATIONAL REQUIREMENTS: Large amount of main memory, also distributed, to enable big data in-memory analytics.

4.3.2.9 Multimember/ Statistical Analysis

This block represents the last stage of the workflow that performs statistical analysis and validation of the feature extraction results, as well as intercomparison of the results from different approaches and models. It consists of two main sub-phases:

- **Statistical analysis and validation**: will perform some statistical analysis on the output of the feature extraction for example to count the number of cyclones per basin or their distribution according to the intensity. Moreover a comparison between the output of the deterministic and the ML-approaches will also be considered as part of this block
- **Multi-model analysis**: will compare the results of the feature extraction block applied on input datasets from different CMIP6 models (considering for example those from HighResMIP) in order to perform an ensemble analysis.

INPUT DATA: NetCDF or textual data from the Feature Extraction stage.

OUTPUT DATA: NetCDF data and maps with the results of the analysis.

COMPUTATIONAL REQUIREMENTS: Large amount of main memory, also distributed, to enable big data in-memory analytics.

4.3.3. Workflow building blocks description

The following table shows the different building (macro) blocks detailing the actions performed, input and output and some implementation details for each of them; specifically it extends the table already reported in D5.1, section 5.3, with additional information on the technological aspects related to the specific tools/frameworks that we plan to exploit in the context of the execution of the workflow.

Table 8 - Statistical analysis and feature extraction workflow building blocks

Building Block	Name	Included actions	Input/Output data structure	Implementations Details
1	CMCC-CM3 simulation run	Run of the CMCC-CM3 climate model	Input: Initial condition, radiative forcings Output: Netcdf format files (~1 TB for 1 year of simulation)	Starting from the Initial Conditions and radiative forcing, the simulation run produces gridded data in netcdf format about the main climate variables. As described in section 3.2 YORC will take care of the initial setup of the workflow. PyCOMPSs has the task to orchestrate the different tasks of the workflow.

2	Pre-processing phase	Concatenation of timesteps, regridding (if needed), variables selection, etc.	Input: CMIP6 or CMCC-CM3 datasets (NetCDF) Outputs: NetCDF files suitable for TC detection/tracking or Analytics blocks	Performs a set of preliminary steps to organize/modify/regrid the data accordingly for the following substeps. PyCOMPSs scripts will be used to manage this task while a combination of dedicated tools for manipulating climate datasets (cdo, nco) along with the Ophidia Framework will be exploited.
3	Feature Extraction	Potential storm identification and storms tracking, computation of different statistical features (e.g. Nr of TCs per basin, distribution in the different categories, etc.).	Inputs: Multiple variables from ERA5 data (3-hourly data spanning from 1979 to 2020 -NetCDF format) needed for ML NN training, IBTrACS observations (historical TC best track data), NetCDF output of the pre-processing phase. Output: NetCDF,txt files, maps with TC detection-tracking and statistical analysis results	Following a deterministic and data-driven approach, extracts TC detection/tracking datasets. In addition, performs statistical features computation on TC related datasets and validation with respect to observations. While PyCOMPSs will take care of the management of the execution, the deterministic approach of TC detection will exploit the TSTORMS tool (see also D5.1, section 5.3). The correspondent TC detection based on a ML approach will exploit the EDDL tool.
4	Multimember/ Statistical Analysis	Percentile/threshold based extreme events indices computation on temperature/precipitation (e.g. heat waves, …), multi-model trend analysis, multi-model intercomparison, etc.	Input: Pre-processed CMCC-CM3 dataset (Netcdf format), statistical analysis from Feature Extraction (Netcdf format), Observational best track data Output: Netcdf, txt files, maps with indices/analytics results data	Performs a Multimember and statistical analysis operations extracting aggregated added values from the climate simulation run or from the Feature Extraction phase outputs. In addition, performs validation with respect to observations. PyCOMPSs scripts will be used to manage this task. If necessary, for simple computations dedicated tools for manipulating climate datasets (cdo, nco) will be exploited. In addition, the Ophidia Framework will be used for the most computational demanding tasks.

5. Technologies & components involved

Starting from the requirement analysis performed in D5.1 (at the level of the pillar) and in D1.1 (at the level of the project software stack) a preliminary mapping of the workflow building blocks with respect to the eFlows4HPC software components is here proposed.

The main programming language that will be exploited for the implementation of these workflows is Python, since it represents a very popular language and many of the solutions supported by the project provide Python bindings.

Table 9 - Mapping of the technologies involved and the building blocks

eFlows4HPC software component	***Workflow block***	***Comment***

PyCOMPSs - COMPSs runtime	General workflow (all the building blocks)	The *COMPSs runtime* will be used to orchestrate and execute the various workflow building blocks though its *PyCOMPSs* interface.
Ophidia HPDA framework	Pre-processing stages, extreme event analytics and multi-member/statistical analysis	*Ophidia* will be the main solution used to implement the pre-processing and data analytics; in particular, the *PyOphidia* interface will be used for coding the block.
EDDL	ML-based TC detection training and inference	*EDDL* will be used for the implementation of the Neural Network used for the ML-based TC detection stage.
Hecuba	Between pre-processing and analytics stages, also for the pruning/diagnostics	The use of *Hecuba* will also be explored to manage intermediate results between the different workflow building blocks
Yorc	initialization/deployment /undeployment stages of the ESM Workflow	YORC will be used for the high level part of the workflow that is based on TOSCA

6. Conclusions

The design process of scientific applications is a very complex task that involves a lot of research and assumptions on the underlying technologies that will be used as base technologies for the implementation phase.

In this document we provide an overall overview of some key aspects that should be contemplated during the development such as which technologies should be used for each part of the ESM workflow and some diagrams to illustrate in a high level way how to exploit the novel eFlows4HPC software stack in order to fulfil the defined requirements and use cases specified in the deliverable D5.1.

This document will serve as a guide for the upcoming development phases.

7. Acronyms and Abbreviations

Term or abbreviation	***Description***
AWICM3	AWI Climate Model Version 3 (OIFS and FESOM2 models)
CA	Consortium Agreement
D	Deliverable
DAG	*Data transformation pipeline*
DLS	*Data logistics services*
DoA	Description of Action (Annex 1 of the Grant Agreement)
EB	*Executive Board*
EC	*European Commision*
GA	*General Assembly*
HPC	High Performance Computing
HPCWaaS	*HPC Workflow as a service*
HPDA	*High Performance Data Analytics*
IPR	Intellectual Property Right
KPI	Key Performance Indicator
M	*Month*
ML	*Machine Learning*
MS	*Milestones*
PM	Person month / Project manager
Pyfesom2	Python based tools to analyze FESOM2 model data
TC	*Tropical Cyclone*
DOI	*Digital object identifier*
UC	*Use case*
WP	Work Package
WPL	Work Package Leader
YORC	Ystia Orchestrator

8. References

[1] Kalman, R. E. (1960). "A new approach to linear filtering and prediction problems". Journal of Basic Engineering. 82 (1): 35–45. doi:10.1115/1.3662552. S2CID 1242324.

[2] Real, R., & Vargas, J. M. (1996). The probabilistic basis of Jaccard's index of similarity. Systematic biology, 45(3), 380-385. doi:10.1093/sysbio/45.3.380

[3] FESOM2 model grid geometry, https://fesom2.readthedocs.io/en/latest/geometry.html#the-placement-of-variables

[4] Zhao,M., I.M.Held, S.-J. Lin, andG.A.Vecchi, 2009: Simulations of global hurricane climatology, interannual variability, and response to global warming using a 50-km resolution GCM. J. Climate, 22, 6653–6678, doi:10.1175/2009JCLI3049.1.

[5] Wehner, M., , Reed, K. A., Stone, D., Collins, W. D., & Bacmeister, J. (2015): Resolution Dependence of Future Tropical Cyclone Projections of CAM5.1 in the U.S. CLIVAR Hurricane

Working Group Idealized Configurations, Journal of Climate, 28(10), 3905-3925. Retrieved Aug 10, 2021, from https://journals.ametsoc.org/view/journals/clim/28/10/jcli-d-14-00311.1.xml

[6] Bacmeister, J. T.,K. A. Reed, C. Hannay, P. Lawrence, S. Bates, J. E. Truesdale, N. Rosenbloom, and M. Levy, 2016: Projected changes in tropical cyclone activity under future warming scenarios using a high-resolution climate model. Climatic Change, doi:10.1007/s10584-016-1750-x

[7] E. Scoccimarro et al., Tropical cyclone interaction with the ocean: The role of high frequency (sub-daily) coupled processes. J. Clim. 30, 145–162 (2017).

[8] Russo, S., Sillmann, J., Fischer, E., 2015. Top ten European heatwaves since 1950 and their occurrence in the coming decades. Environ. Res. Lett. 10, 124003. doi: 10.1088/1748-9326/10/12/124003

[9] Dirk Barbi, Nadine Wieters, Paul Gierz, Miguel Andres-Martinez, Deniz Ural, Fatemeh Chegini, Sara Khosravi, and Luisa Cristini (June 2021). ESM-Tools version 5.0: a modular infrastructure for stand-alone and coupled Earth system modelling (ESM) Geoscientific Model Development 14(6):4051-4067 doi:10.5194/gmd-14-4051-2021

Appendix A.

ESM-Tools

ESM-Tools is a set of software components designed to provide a common framework for handling the most typical tasks to run an earth system model successfully such as downloading and compiling the model components, running the model either coupled or standalone and performing data diagnostics on the output data.

ESM model setup utils

Conventional workflow on an HPC using an ESM simulation involves: compiling the model(s) on target architecture, linking with external, dependent libraries (optionally also compile), specifying inputs (optionally build them) and experiment parameters in namelists, and submit the experiment to a job scheduler for execution. To facilitate the use of an ESM in envisioned modular and deployable workflow architecture of eFlows4HPC requires clear separation of tasks that are configurable using a specification (eg., in YAML) that can be easily shared and used by other tools of the workflow.

OpenIFS/FESOM2 is designed to be used with a helper software package, esm_tools [9]. Fortunately, goals of esm_tools are congruent with specified modular requirements of the workflow described in the previous sections. The esm_tools, essentially a Python package, includes specifications, in YAML, for supported models and HPC configurations to compile, configure and perform standard experiments. These tasks are mostly achieved by the following packages, using OpenIFS/FESOM2 as model:

- **esm_master**: handles fetching source code for independent components of our ESM model, namely: OpenIFS (atmospheric model), FESOM2 (ocean model), and OASIS coupler, and their compilation on the underlying HPC platform. It has additional options such as to elegantly handle updates to source codes and versions.
- **esm_runscripts**: takes runtime specific configuration specified in YAML as input to execute an ESM simulation. A typical ESM can have multiple namelists (for underlying models) that often contain many variables. The input YAML is hierarchically composable based on an elaborate default base configuration. In effect, this simplifies and reduces clutter in specifying input; only variables changed from base config are required, defaults are inferred automatically. This also means input specification can be hashed hierarchically, thus promoting simple and reproducible experiment specification.

While ESM-Tools provide a useful starting point, we intend to contribute to its development to at least accommodate the needs of this project. Desirable features include: use TOSCA specification as machine specification instead of its non-standard specification, extend functionality to accommodate inputs from a variety of data sources (eg., using APIs provided by data logistics service) instead of currently used, hard coded file paths.