eFlows4HPC
Enabling dynamic and Intelligent workflows in the future EuroHPC ecosystem

D5.1 Requirements on the eFlows4HPC software stack from Pillar II and evaluation metrics

Version 1.0

Documentation Information

Contract Number	9555558
Project Website	www.eFlows4HPC.eu
Contractual Deadline	30.06.2021
Dissemination Level	PU
Nature	R
Author	Nikolay Koldunov (AWI)
Contributors	Alessandro D'Anca (CMCC), Julian R Berlin (BSC), Donatello Elia (CMCC), Enrico Scoccimarro (CMCC)
Reviewer	Steven Gibbons (NGI)
Keywords	Workflow's description, requirements, metrics

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

Change Log

Version	Description Change
V0.1	Proposed ToC
V0.2	Ready for internal review
V1.0	Ready for submission

Table of Contents

1. Executive Summary

This document presents the work performed in WP5 regarding gathering and systematizing information about typical Earth System Model (ESM) workflow building blocks and their software and infrastructural requirements. We also define relative metrics to measure improvements and success of the eFlows4HPC workflow development relating to the Pillar II use cases.

First, a detailed description of the typical ESM workflow steps is provided, followed by description of innovative workflow components developed within Pillar II (namely dynamical data analysis and feature extraction). From those descriptions, we draw requirements for the software stack and infrastructure that have to be matched during development of the eFlows4HPC architecture. As the final step, we select metrics to measure different quality aspects that are considered important to take into account during the design process that will be followed. We describe the applicability and importance of those metrics to the ESM workflow.

2. Introduction

The atmospheric, ocean, and later Earth System Models simulations on HPC have a long history, dating back to the 1960s. This is one of the most challenging HPC use cases, not only due to very high computational cost, but also due to additional challenges related to intensive I/O patterns, very large data volumes, and the necessity of not only producing data on HPC, but also post-processing it there.

The aim of this document is to support development of the eFlows4HPC architecture by providing the following information:

- **Clear description of the basic workflow building blocks.** This will allow workflow developers to get a better understanding of what the typical ESM workflow consists of.
- **More detailed description of innovative workflow components.** As part of the eFlows4HPC tasks is to develop innovative components of the workflow, that will help to save resources and allow performing of more ambitious scientific tasks, the steps involved in execution of those components should be described in more detail.
- **Define software and infrastructure requirements of the workflow and its components.** We will use information about basic steps involved in the workflow in order to determine those requirements. The team developing the general eFlows4HPC architecture can then match those requirements with software from the eFlows4HPC stack and other relevant solutions.
- **Define metrics to evaluate the resulting eFlows4HPC workflow.** We will provide metrics that are relevant to Pillar II with description on how ESM workflows will benefit from improvements in those metrics.

This document sets the stage for further development of the general and Pillar II-specific elements of the workflow and defines criteria for evaluation of progress in this development.

3. General description of ESM workflow and its components

The classical climate modelling workflow consists of several preparation and post-processing stages that we are going to outline below. First, we present a list of main steps in an order that is close to the chronological order during the execution of the workflow with innovative steps proposed by eFlows4HPC marked by bold font.

- 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
    - **Dynamical data analysis**
    - Data output
- Data post-processing
- Data analysis
    - Sanity checks/ standard diagnostics
    - Scientific analysis of the data
    - **Feature extraction**
- Data archiving
- Data distribution

Preparation of model computational mesh

Computation of most of the components of the climate models happens on a computational grid (or mesh). It splits some geographical domain (the whole globe, ocean, land, particular area) into small regions that are represented by discrete points, that can be either aligned regularly (quadrilateral grid), or irregularly (unstructured mesh). For many Earth system components there is also vertical discretization. The users have to decide on the configuration of the mesh taking into account the balance between the lateral/vertical resolution necessary to resolve processes they are interested in and the computational resources one has to spend. **In most cases computational meshes are reused**, as every new mesh most probably means a new set of tuning parameters. However, there are cases when mesh generation could be part of the workflow. It is usually in connection with situations when users want to have better resolution in some particular region, motivated either by scientific needs or urgency of the situation (oil spill, natural disaster). Since components of the climate model often have different computational meshes, part of the mesh preparation is **generation of interpolation weights that** allow a quick interpolation of information that has to be exchanged between model components. An additional task is partitioning of the grid, to distribute model domain among different computational cores. This is trivial in the case of regular grids that cover the whole globe but requires some additional software (partitioner) if the grid only partly covers the Earth’s surface, or is unstructured.

Preparation of initial conditions

The components of the climate model should have some initial state that is characterized by a distribution of properties, such as temperature and salinity in the ocean, or atmospheric pressure in the atmosphere. There are two main sources, where spatial distributions of those properties can be obtained. One is observational data, usually taken in some gridded form for the area of interest (e.g. global atmosphere, or ocean). Another is the results of simulations performed by other models, often performed on different computational grids. The initial conditions data required by the user should be available on the file system and interpolated to the computational grid. Sometimes interpolation functionality is built into the model, but in most cases some adjustments of the data format are necessary. In case of the ensemble runs, perturbations in initial conditions are usually used to make ensemble members slightly different from each other. The basic requirement from the workflow would be to provide information on where different forcing files are located (Catalog), how to connect them to the model setup, and how to interpolate them when needed.

Preparation of forcing data

For climate models, initial conditions alone are not enough, there should be periodic updates of information about conditions not directly simulated by the climate model or one of its components. In the case of the full coupled model examples are solar constants, CO2 concentrations, and aerosol concentrations. This additional flow of information is called forcing. For a climate model, usually the amount of this information is not large (as it may contain most of the components generating the fluxes), but things change when components of the climate system run separately. For example, if an ocean model runs in a stand-alone regime, there should be constant updates (at least several times per model day) on the information about surface atmospheric temperature, wind, radiation and so on. This information may correspond to actual time series, or just climatic averages, and it is taken from pre-processed files that can be large in volume. The basic requirement from the workflow would be to provide information on where different forcing files are located (Catalog) and how to connect them to the model setup.

Model compilation

The code of the model should be checked out from the repository and compiled for the particular machine. As coupled models sometimes consist of components that are located in different repositories, the workflow should make sure that the right versions of the code that are compatible with each other are checked out. Compilation settings depend on the machine and libraries/compilers the user would like to use. This information should be available either at the model level, or somehow controlled by the workflow.

Preparation of model configuration

Climate models naturally can have different setups that differ by grid configurations, initial conditions, and types of forcing, as well as by parameters of dynamical and parameterization packages. The settings are changed according to the scientific tasks, but are also heavily dependent on the model grid and input files. The proper default parameters will depend on a combination of grid, initial and forcing conditions that users have selected. The task of the workflow is to select this proper combination and generate configuration files for each component of the climate model, as well as for the whole coupled setup. In case of the ensemble runs, configurations should be prepared for each ensemble member. The files necessary to submit the job to an HPC also have to be generated. An additional task at this stage would be to save as much information as possible about model code, model configuration, and computational environment to foster reproducibility of the results and make bug-tracing easier.

Model run/monitoring

The model run(s) (or runs in the case of an ensemble) should be submitted to a supercomputer, and the monitoring of the job state should be performed regularly. Note that in the case of ensemble runs, each instance of the climate model usually has a different runtime directory, in which all the outputs, logs and model checkpoints/restarts will be saved. The initial data that is common to all the ensemble members can be just linked from a common directory, but there are some other files which belong to each instance (sometimes perturbed copies of a common one, as explained above) and need an individual copy per ensemble runtime. There should be a possibility to resubmit the run for cases where the allocated time is over or some simple error has occurred and notify the user about the status of the job and help to find errors.

Data output

During the model run the usual procedure is to dump from time to time the state of the model to disk into so-called restart files. As climate simulations usually take days or even months to complete, they have to be computed in chunks that fit to HPC job allocation requirements. Moreover, it allows you to modify parameters (e.g. time step) if the model becomes unstable and rerun the chunk of the model from this saved intermediate state.

Model output/diagnostics (in terms of arrays on the computational mesh) are dumped to disk during the model run, so that the user can analyze the data afterwards. If the data are not used for monitoring of the model run, already at this stage they can be archived. The volume of the data and the frequency of the output depend on computational grid and user settings and can vary from gigabytes to terabytes per model year. There are different types of model I/O implemented in climate models, starting from sequential single core to asynchronous output that happens without interrupting model computation.

Dynamical data analysis

Information about changes in the running model state can be accessed directly from memory, without serialization to the disk, and analyzed asynchronously, parallel to the model run. This solves the problem of *simulation/diagnostics* I/O bounds, since saving data to disk is a slow operation, and for some types of *data analysis/model resolutions/number of ensemble members* it can’t be performed in practice with high enough frequency.

Data post-processing

Usually data that are outputted by the climate models are already in the format that can be directly used for data analysis. However, for some of the variables conventions have to be made in order to put data in more convenient coordinates, or units. In the case of large intercomparison projects, like the Coupled Model Intercomparison Project (CMIP), model data should be provided in the specific format with predefined sets of meta-data, requiring conversion. The latter stage can be combined with archiving (see below).

At this point the data analysis part begins, and here the possibilities are limitless. The workflow should provide the user access to the data stored on disk and the possibility to process those data with different tools. This can be either general purpose tools, like programming languages (e.g. Python, R, Matlab) and libraries for those languages (numpy, scipy, xarray), or special programs for working with climate model data (e.g. cdo, nco). In addition, there can be predefined post-processing pipelines, aimed at some specific analysis. One of such examples is the **Feature extraction workflow**, described in the chapter below. Another is a predefined set of sanity checks/standard diagnostics that are performed for each model run. As for every climate model this set of diagnostics will be different, the general workflow probably only has to provide an interface for calling that diagnostic and providing an execution environment and necessary resources.

Depending on the amount of data and computational complexity of the data analysis, the post processing requires a large range of computational resources. It spans from sequential analysis on one core with a small amount of RAM, to the multi-node parallel HPC jobs, running for days. The range of the data volume produced by those analyses, and hence the I/O and disk space requirements are also quite large. It can be a time series of 100 records, characterizing the mean global atmospheric temperature for each year of the century, or derived variables (e.g. vorticity, total precipitation), that have the same size as the original data.

Most of the data processing scenarios in climate science require that data are available on the disc. There are options when copying chunks of the data from a tape archive, performing analysis on them and then proceeding to the next chunk, if possible, but they are rare and inefficient.

Data archiving

For key experiments, like those used for publications, intercomparison projects, or multiple research projects, data archiving has to be performed. Usually the data is copied to the tape archive to be stored for long times and to be retrieved, when additional analysis is necessary. In some cases this step is combined with model run and data post-processing, when data are post-processed and archived with some frequency (e.g. each 5 model years). For this step it is important to have well defined mechanisms/policies of data archival and retrieval, so all the necessary data are archived and can be easily retrieved by request.

Data distribution

In some cases wide access to the data should be provided for users that do not have access to HPC archives. Most prominent examples are data that is used for preparation of scientific publications, and results of large coordinated computational efforts (CMIP), that should be available for multiple groups around the world. Usually services that provide capabilities for such data distribution also provide Digital Object Identifiers (doi) and guarantee that for some time data will be available. Currently, locating data in the cloud has become popular, which allows to “bring the code to the data”, and perform data analysis using cloud resources instead of downloading large amounts of data to local computing centers.

4. Innovative components of the Pillar II workflow

The overarching goal of WP5 is to take advantage of the eFlows4HPC architecture to enhance innovation for intelligent and integrated end-to-end HPDA-enabled ensemble Earth System Model (ESM) workflows.

A typical ensemble workflow experiment consists of several start dates and members divided in several sequential simulation chunks. A strong and performant workflow solution is needed since these types of experiments are very resource consuming, due to their static nature. It is very hard to have a flexible approach.

On the other hand, data-driven solutions can provide new methodologies for analytics and feature extraction at scale, for example with respect to multi-model analysis and extreme event analysis.

Figure 1 shows an overview of the general workflow targeted by eFlows4HPC Pillar II on climate.

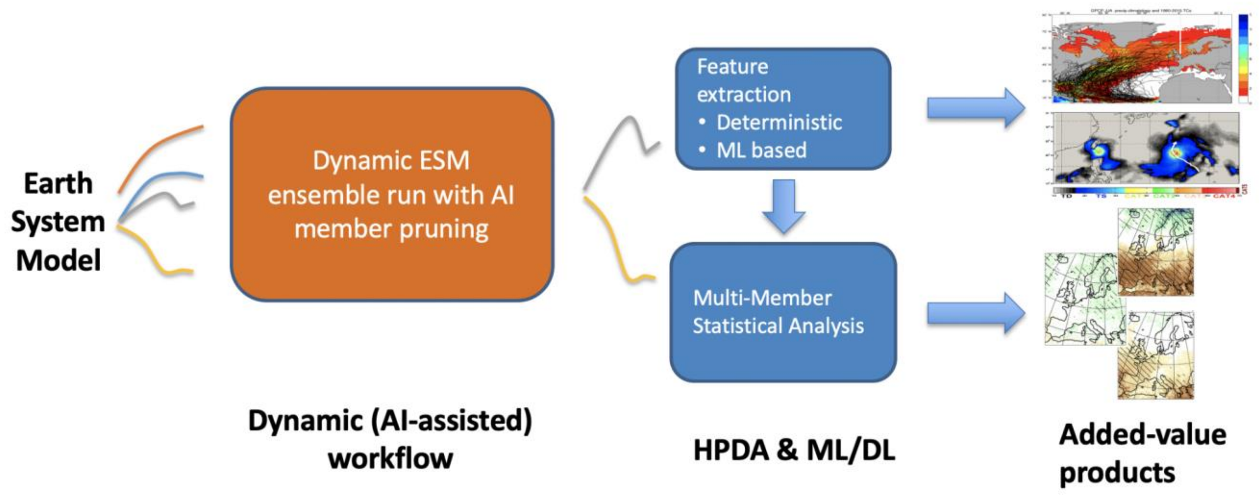

Earth System Model
Dynamic ESM ensemble run with AI member pruning
Feature extraction
• Deterministic
• ML based
Multi-Member Statistical Analysis
Dynamic (AI-assisted) workflow
HPDA & ML/DL
Added-value products

Figure 1. General Pillar II workflow

4.1 AI-assisted ESM member diagnostic component

The overall idea is that eFlows4HPC provides components or functionality to allow ESM simulation runs the capacity to prune members that don't add anything useful to the whole simulation. The idea is to make a better use of computational and storage resources by performing a smart (AI-driven) pruning of ensemble members (and releasing resources accordingly) at runtime.

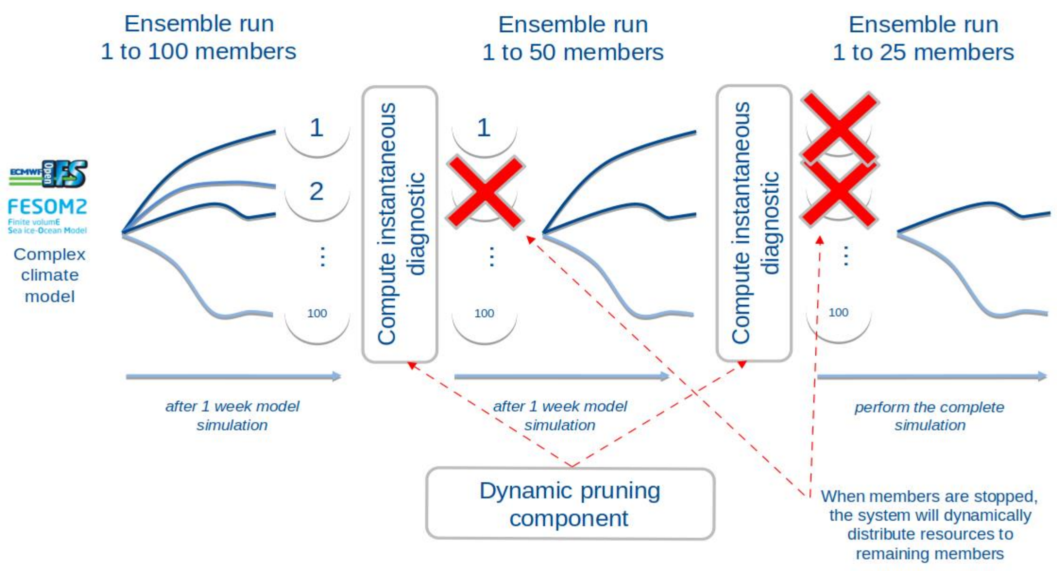
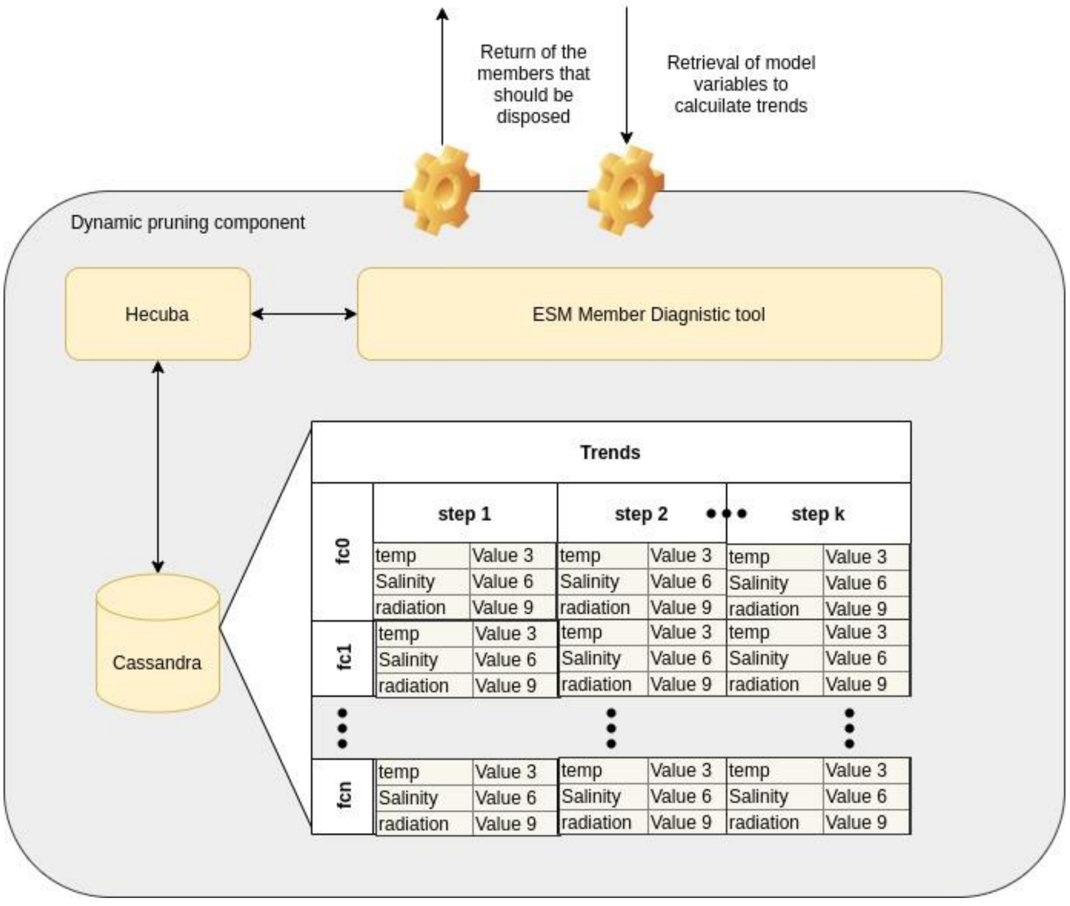

Ensemble run
1 to 100 members
Ensemble run
1 to 50 members
Ensemble run
1 to 25 members
FESOM2
Finite volumE Sea ice-Ocean Model
Complex climate model
1
2
100
Compute instantaneous diagnostic
after 1 week model simulation
perform the complete simulation
Dynamic pruning component
When members are stopped, the system will dynamically distribute resources to remaining members

Figure 2. Dynamic AI-assisted workflow

A possible initial approach could be to take pruning decisions on variables and accumulated metrics that may vary over time and that we are able to store and check during the different time steps of an ensemble run, for each defined member in the ensemble.


Return of the members that should be disposed
Retrieval of model variables to calculate trends
Dynamic pruning component
Hecuba
ESM Member Diagnistic tool
Cassandra
Trends

	step 1		step 2		step k	
fc0	temp	Value 3	temp	Value 3	temp	Value 3
	Salinity	Value 6	Salinity	Value 6	Salinity	Value 6
	radiation	Value 9	radiation	Value 9	radiation	Value 9
fc1	temp	Value 3	temp	Value 3	temp	Value 3
	Salinity	Value 6	Salinity	Value 6	Salinity	Value 6
	radiation	Value 9	radiation	Value 9	radiation	Value 9
fcn	temp	Value 3	temp	Value 3	temp	Value 3
	Salinity	Value 6	Salinity	Value 6	Salinity	Value 6
	radiation	Value 9	radiation	Value 9	radiation	Value 9

Figure 3. Overview of the Dynamic ESM pruning component

Then we will conduct an assessment about similarity in which two thresholds are defined, either to say the member is an outlier or is essentially identical to at least one of the ongoing members that are currently executing in the ensemble run. One of the challenges here is to define the initial starting point for conducting the pruning; if we do this too early, we may discard members that actually will produce useful results for the simulation as a whole. This way we can have a dynamic workflow with optimized resource usage.

4.2 Statistical analysis and feature extraction

The eFlows4HPC infrastructure will be also exploited in the context of the case study related to the multi-model analysis of a Tropical Cyclone (TC) track. Figure 4 provides a high-level view of the workflow considered.
In order to evaluate how TCs activity might change under different climate conditions - in terms of landfall, associated strong winds, heavy precipitation etc. [1,2] - it is important to investigate their representation by General Circulation Models (GCMs). Also, our knowledge of the TC interaction with the climate system can build on GCM results [3-5]. This can be done following different detection and tracking methods [6] available in literature and also investigating new Machine Learning approaches, to verify the possibility of speeding up the detection process in the context of a multi-model multi-member analysis.

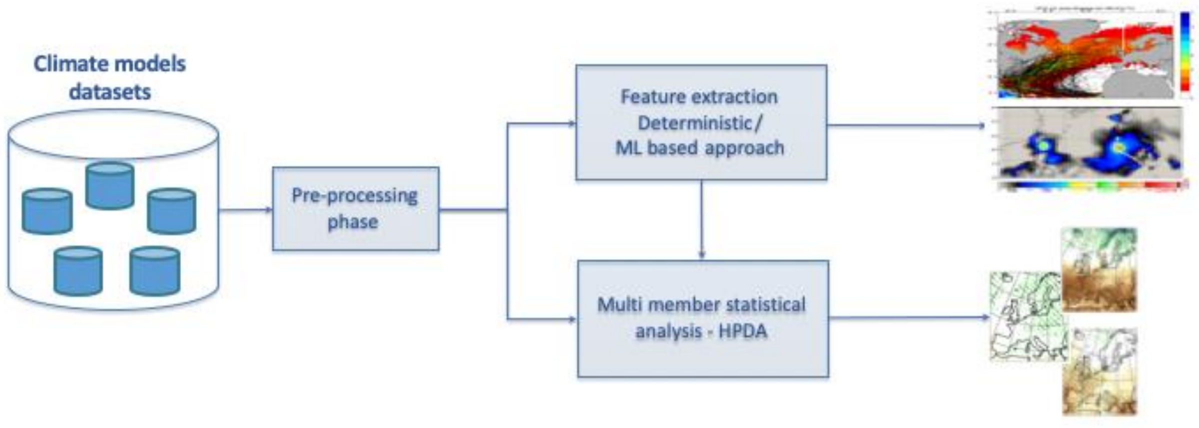

Climate models datasets
Pre-processing phase
Feature extraction Deterministic/ ML based approach
Multi member statistical analysis - HPDA

Figure 4. Feature extraction workflow

The case study joins together (i) the tropical cyclone detection and tracking on three-dimensional fields, like pressure, temperature, wind velocity, and vorticity and (ii) the multi-model analysis on the products resulting from the first step. The analysis will take into account two different use cases based on different input sources and considering different application scenarios.

The first one represents the base scenario where the extraction and further analysis of TC detection and tracking information are applied on the output of a single climate model, specifically the CMCC-CM3 model. CMCC-CM3 is the latest model version under development at CMCC, based on the previous version of the CMCC coupled climate model [7,8] and 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. The important and strategic difference with the NCAR coupled model is the oceanic component. As a member of the Nucleus for European Modelling of the Ocean (NEMO) consortium (https://www.nemo-ocean.eu), CMCC takes an active part in the development and evolution of ocean related engines. That motivated us to replace the CESM ocean component with the NEMO physical core. Based on the

CMCC-CM2 model, CMCC contributed to the Coupled Model Intercomparison Project phase 6 [9]. The model is an essential component of the CMCC seamless simulation system that spans operational seasonal predictions and climate scenario productions. In CMCC-CM3, the atmospheric component is the CAM6 and the ocean component is NEMO 4.0.

The second scenario will consider large volumes of data from centennial CMIP6 products at the highest horizontal resolution available (e.g. ¼ degree resolution) and with a temporal frequency of at least 6 hours: by coordinating the design and distribution of global climate model simulations of the past, current, and future climate, the Coupled Model Intercomparison Project (CMIP) has become one of the foundational elements of climate simulations. In this case CMIP6 experiments outputs will feed the TC detection and tracking procedures and the subsequent multi-model multi-member analysis.

In general terms, the overall case study will perform an in-depth comparison (by means of specific statistical analysis and scientific validation approaches) between deterministic and ML-based schemes (e.g., based on Neural Networks) for TC detection and tracking by leveraging the eflows4HPC infrastructure, workflow approaches, HPDA and ML/DL solutions. More in detail, HPDA frameworks (e.g. Ophidia) will be used to exploit HPC architectures and parallel (MPI/OpenMP) environments to perform statistical analysis, intercomparison operations, indices computation from large amounts of data, for instance in a multi-model ensemble analysis perspective. Parallel analysis, which allows the performance of complex operations on multiple computational cores or concurrently on different sets of data, and parallel I/O, to speed-up the retrieval/storing of data from/to the underlying storage, are key features to take into account in the design of the data analytics approach in the Pillar II context.

5. Requirements & Constraints

In order to facilitate development of the general workflow architecture, in this chapter we will provide an overview of the software requirements and constraints coming from typical tasks that have to be performed in the ESM related workflows. First a general overview of the workflow building blocks with associated actions, input/output data structures, indication of HPDA/ML resource usages, deployment location and typical time scales will be given. Then more concrete software and infrastructure requirements to perform those actions will be listed and discussed. We will do it first for the generalized Pillar II workflow, and then in more detail for innovative components developed in the framework of eFlows4HPC.

5.1 Requirements from general Pillar II workflow

In this section we list common requirements that are applicable for general purpose ESM workflows. To begin with, below is the table where main building blocks of the typical ESM simulation and data processing workflow are presented. This should help to identify software solutions suitable for each of the building blocks. The building blocks cover steps from preparation of different types of input data for running the ESM, through model execution and data post-processing to final archiving of those data in a long-term storage or distributing it to the public.

Table 1. Building blocks of the typical ESM simulation and data processing workflow

Building Block	Name	Included actions	Input/Output data structure	HPDA/ ML	Deployment	Time scale	Description
1	Preparation of model computational grid	Discovery of data location. File copy, grid partitioning if needed.	Input: mesh files, ASCII/netCDF Output: mesh files, ASCII/netCDF, grid partition.	NO	HPC	some seconds/ hours	The grid should be copied from a remote location or linked if data is already available. Partitionings can be precomputed and stored in a data pool.
2	Preparation of initial conditions	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)	NO	HPC (login nodes)	some seconds/ minutes	The initial conditions should be copied from a remote location or linked if data are already available.
3	Preparation of 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.	NO	HPC (login or compute nodes)	Depends on the size, up to few days	Usually this operation is performed only once for each of the computational grids. Most optimal is to have data already prepared in the data pool.
4	ESM model compilation	Compilation of all model components, linking libraries.	Input: model source code. Output: model executable.	NO	HPC (login or compute nodes)	Minutes	Require information about available compilers, libraries and system configuration.
5	Preparation of ESM model configuration	Creation and modification of configuration files. Retrieving missing data. Job definition	Input: general configuration files. Output: configuration files adjusted for specific model simulation.	NO	HPC (login or compute nodes)	Minutes	Users define parameters of experiment that have to be converted to proper configuration of the model considering also information about the infrastructure the model will be run on and available resources.
6	ESM model execution	Execution of the model on available resources (see also 5.2 and 5.3). Monitoring of the results, and execution. Online data analysis (see 5.2). Resubmission of jobs.	Input: grid, initial conditions, forcing (netCDF, ASCII) Output: model results (netCDF, ASCII, GRIB), check points, monitoring results.	Sometimes	HPC	Hours/days	Model execution can contain several subtasks, that can involve monitoring and online data analysis (see also 5.2 and 5.3)
7	Initial data post-processing	Conversion of data to different formats for further analysis.	Input: model data on the disc (netCDF). Output: post processed data transferred to disc partition where analysis	YES	HPC, cloud, GPU	Hours/days	The final result is the data fully prepared for further scientific analysis. Can involve a fair amount of DA tasks

		Deriving additional variables Adding metadata Data transfer to disc partition where they can be analysed. Initial archiving	can be performed (netCDF, zarr)				and data transfers. See also 5.3.
8	Data analysis	Initial sanity check Running standard diagnostics Interactive/expl oratory DA Scientific analysis (e.g. Feature extraction, see 5.3)	Input: Post-processed data (netCDF, zarr) Output: Results of data analysis as data files netCDF, ASCII or images.	YES	HPC, cloud, GPU	Minutes/da ys	There are many types of DA that are performed with climate model data, some of them are workflows in their own right (see for example 5.3).
9	Data archiving/distribu tion	Copy data to archive Record information on where data can be found and how they can be retrieved. Add DOI Expose data for users outside HPC	Input: Post-processed data (netCDF, zarr) Output: Same data, but located in the archive.	NO	HPC	Minutes/da ys	This only applies to data that can be useful for further analysis.

Although different components of ESM workflows have variable requirements in terms of software and infrastructure, we first list minimum requirements that are common for every ESM workflow. The architecture of the workflow should provide access to the listed software and make sure that resources in the infrastructure are available and can be allocated.

Programming languages

There is a standard list of HPC-oriented programming languages that should be available on the system in order to be able to compile ESM models and execute supporting scripts. The workflow should provide information to workflow components on availability and details of compiler installation (path to compilers, compiler names, etc.).

Table 2. Standard programming languages necessary to compile components of the typical ESM workflow.

Programming language	Versions/distributions	Role in the ESM workflow
sh	bash, csh/tcsh, ksh	Scripting at all stages of workflow execution
Python	conda installation with possibility to install user software	Scripting, data processing and visualization, glue language between different components.
FORTRAN	Commercial versions. Usually Intel works best.	Compilation of model code and analysis tools
C++	Can be gcc or commercial versions	Compilation of model code and analysis tools

Specialized software dependencies

There are several software libraries that have become de-facto standards in the ESM community, and should be available for any version of ESM workflow. The workflow should provide information to workflow components on availability and details of library installations (path to libraries, options they are compiled with, e.t.c).

Table 3. Standard libraries necessary to compile components of the typical ESM workflow.

Library	Versions/distributions	Role in the ESM workflow
MPI	Commercial versions or OpenMPI.	Parallelization. The workflow should allow model compilation with different versions of the MPI libraries, e.g. IntelMPI, OpenMPI, if the ESM has this option.
OpenMP	Commercial versions or free.	Parallelization. Support for OpenMP parallelisation is not as common in ESM components as MPI, but still can be found quite often.
BLAS/LAPACK	Commercial or free versions. Intel MKL.	linear algebra libraries. The popular implementation used together with Intel Fortran compilers is from Intel oneAPI Math Kernel Library.
netCDF FORTRAN/C	Free versions compiled by different compilers.	The netCDF is the most popular, and de-facto standard data format for climate simulations. Most of the ESMs are capable of outputting data in netCDF format and need netCDF libraries for compilation.

Infrastructural requirements

The workflow should provide information to workflow components on what infrastructure is available on the machine, how to use it, and check if there are enough resources that can be allocated to execute a particular configuration of the workflow. The most important resources are the following:

- **Availability of computational resources**. The number of cores used by ESMs depends on the size of the task and can range from just a few cores for test configurations to hundreds of thousands of cores, when very high spatial resolution is used. Post processing tasks usually require an order of 100 cores if the post processing is done in parallel.
- **Availability of fast networking communications**. Often computations in ESM components are bound by communications (e.g. Koldunov et al., 2019) and hardware that allows for faster communication is better suited for running ESM models.
- **File system**. The ESM model simulations produce large amounts of data, and for some setups (e.g. stand-alone ocean or atmosphere simulations) also consume considerable amounts of data. That's why it is important to have a fast and responsive file system that will not slow down the simulations during I/O operations. The I/O operations are also often

a bottleneck for post-processing tasks, especially those related with reading a lot of data into the memory.

- **Online storage**. The large data volumes produced by ESM simulations should be stored on disk for a relatively long time before the post processing is finished and the data are ready to move to archive or deleted. This is why there is a considerable amount of storage (order of at least tens of TB) required for comfortable execution of ESM related workflows. Tasks related to post-processing of multimodal ensembles (like CMIP6) might require even larger amounts of disk space.
- **Offline storage/Tape archive**. Storing all data generated by ESM model simulations in online storage for a long time is not feasible due to large volume, and decrease in data processing intensity with time. Usually the data that should be further analyzed in the future are stored in the offline storage facilities, like tape archives. The workflow should be able to integrate with such a storage and support putting model fields to archive immediately after the simulation (for fields that are not frequently used), as well as after some time the simulation is finished.

5.2 Requirements from the AI-assisted ESM member diagnostic component

This section describes the requirements for the AI-assisted ESM member diagnostic component of the workflow needed to make the ESM workflow dynamic. The considered scenarios are the following:

- Use case 1:
  - This use case describes the main ESM workflow implementation with FESOM2 and OpenIFS models, fully described in the previous section 5.1
- Use case 2:
  - This use case describes the Dynamic ESM part for the workflow previously described
  - The general workflow manager will decide which members to discard based on the result of the assessment done by the AI-assisted Member diagnostic component on the model variables for each running member and do the pruning.
  - A persistent storage solution will be used, as well as Python as a development language.

In principle, both the general workflow manager and AI-Assisted component can run asynchronously.

Requirements from the software functionality

In this section we list a summary of the Dynamic Member diagnostic component building blocks as a part of the ESM workflow requirement specification.

Table 4. Dynamic Member diagnostic component building blocks.

Building Block	Name	Included actions	Input/Output data structure	HPDA/ML	Deployment	Time scale	Description
1	Initialization of the ESM Member Diagnostic component	the diagnostic component prepares DB for the run	Input: Member data, maybe some initial values from the initial conditions Output: N/A	NO	HPC	some seconds	At the beginning of the execution, ESM Member diagnostic component for the ensemble to run is instantiated, a threshold is defined as the point of start to actually begin with the analysis that later will lead to the decision on which members should be discarded.
2	Workflow manager retrieves data from the model	call to Hecuba or ESM member diagnostic interface to store model data	Input: experiment/Ensemble ID, list of variables to retrieve Output: array with variable information and other model related settings	NO	HPC	some seconds/minutes	During the execution of the ensemble, the workflow manager retrieves model diagnostic data and stores it in the Hecuba object repository. It can be at the end of a given chunk or set of chunks.
3	The ESM member diagnostic component conduct the assessment of the executing members data	Data is retrieved from Hecuba database to conduct the assessment	Input: experiment id Output: array with the member that should be discarded before submitting the next chunks of the simulation	NO	HPC	some seconds/minutes	The ESM member diagnostics tool retrieves the accumulated data for the ongoing run and proceeds to calculate trends and deviations on all members currently executing. The information about which members should be discarded is also stored in Hecuba. The approach to do this may involve AI techniques. This is conducted asynchronously periodically. The component determines which members should be discarded and saves this information in Hecuba. It is expected to use some AI techniques to improve this feature.
4	Workflow manager discard members	based on the assessment, members are removed from the ensemble and all its associated data removed	Input: experiment id, member list (array) Output: boolean flag to indicate the removal was successfully achieved	NO	HPC	some minutes depending on the nr of members and associated data, deletion can be done asynchronously	The workflow manager , before submitting to run any new chunk, checks the available diagnostics for the on going members to determine which are the members that will continue executing, for the ones that should be cancelled, the workflow manager executes a disposal of all data generated by the cancelled members in order to free space and mark these as invalid so

							these will be removed from the ensemble in the successive steps. The software stack may provide some form of mechanism for cleaning up data in the middle of a simulation for the cancelled members.

Besides the general description provided in the building blocks table, it is important to mention that the ESM Dynamic workflow depends on specific software from the climate science domain which is not directly part of the eFlows4HPC software stack, specifically the **OpenIFS** and **FESOM2** models, used to conduct climate simulations. The tasks related to the building blocks mentioned in the general case (the use case 1) will be performed by the Integrated workflow manager, as will the orchestration of job submissions/re-submissions, ensemble pruning and monitoring. The ESM member diagnostic component will provide the necessary data analysis on the fly and feed decisions on the fate of ensemble members to the workflow manager.

The following table provides an overview of the data used or produced within the two use cases mentioned above in terms of data format and usage.

Table 5. The format and usage of the data produced by use cases 1 and 2 (section 5.2).

Data sets	**Data Flow**	**Persistency**	**Data type/ structure**	**Consumption pattern**	**Notes**
OpenIFS/ FESOM2 input data	Sources	Persistent	NetCDF/Mesh files	1 to 1 (-> BB#1)	Grids, initial conditions, forcings, etc.
OpenIFS/ FESOM2 output/ restarts	Intermediate	in-memory	NetCDF	1 to 1 (-> BB#1)	intermediate data generated by the model
Diagnostics for member pruning	Intermediate	In-memory	ND arrays	1 to 1 (-> BB#1)	ND Array with diagnostic information on the running members

5.3 Requirements from analysis and feature extraction component

This section describes the requirements for the statistical analysis and feature extraction part of the workflow.

In terms of task flow, the analysis and feature extraction component is formed by 4 different macro-modules: CMCC-CM3 Simulation Run, Pre-processing phase, Feature Extraction and Multi-member/Statistical Analysis. Each of the aforementioned components interacts with one or more modules in terms of Input/Output or required functionalities to perform their specific operations/computations. In addition, each macro-module is composed of different sub-modules able to carry out a single task/analysis/procedure. Following the two main scenarios presented in 4.2, their related flow is described below:

- Use case 1:
  - The workflow starts the execution of the CMCC-CM3 model (building block 1). The ESM simulation runs in parallel with the Pre-processing (building block 2) and the Feature extraction (building block 3) phases;
    - The pre-processing phase (building block 2) prepares the input for the subsequent feature extraction - TC detection and tracking and statistical analysis - phase (building block 3) along with observational data;
  - A further analytics block (building block 4) computes extreme event indices directly on model output;
  - For each iteration of the model execution, the aforementioned steps are executed at model runtime. Input data to the model must also be available on the storage;
- Use case 2:
  - The workflow starts directly from the model data (i.e., CMIP6 dataset);
  - The pre-processing phase (building block 2) prepares the input for the subsequent feature extraction - TC detection and tracking and statistical analysis - phase (building block 3) along with observational data;
  - A final block performs the multi-model analysis (building block 4) over the results from the feature extraction phase (building block 3);
- In both cases:
  - the software components in the various stages (preprocessing, TC detection and tracking, statistical analysis) can be executed when required during the workflow;
  - The ML model used in the Feature Extraction phase (building block 3) should be pre-trained. Training should be executed offline on GPUs using ERA5 and IBTrACS data.

The key aspects of each building block of the workflow presented in this description are summarized in the following table. The templates used to derive the requirement for each building block are enclosed as an Appendix at the end of this document.

Table 6. Analysis and feature extraction component building blocks.

Building Block	Name	Included actions	Input/Output data structure	HPDA/ ML	Deployment	Computational Granularity	Description
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)	NO	HPC (MPI)	3 hours to simulate 1 month	Starting from the Initial Conditions and radiative forcing, the simulation run produces gridded data in netcdf format about the main climate variables
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	YES	HPC	some seconds/ minutes	Performs a set of preliminary steps to organize/modify/regrid the data accordingly for the following substeps.
3	Feature Extraction	Potential storm identification and storms tracking, computation of different	Inputs: Multiple variables from ERA5 data (3-hourly data spanning from 1979 to 2020 -NetCDF format)	YES	HPC - GPU	some seconds/minutes (training	Following a deterministic and data-driven approach, extracts TC detection/tracking

		statistical features (e.g. Nr of TCs per basin, distribution in the different categories, etc.)	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			can require minutes)	datasets. In addition, performs statistical features computation on TC related datasets and validation with respect to observations.
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	YES	HPC	few seconds/ minutes	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.

Besides the general description provided in the table 6, it is worth mentioning that the workflow depends on some specific software components from the climate science domains which are not directly part of the eFlows4HPC software stack, but that can benefit from the integrated workflow approach proposed in the project. The main software dependencies are:

- **CMCC-CM3** - ESM that will be used for development of this component of the workflow. It is noteworthy that the model can run only the CMCC HPC infrastructure. It is written in Fortran and bash scripts and requires MPI and other libraries (e.g. NetCDF and numerical libraries);
- **TSTORMS** - specific tool for TC detection/tracking (eg. https://www.gfdl.noaa.gov/tstorms/);
- **NetCDF-oriented tools**: CLI (e.g., CDO, NCO, Ncview) or Python modules for pre-processing of NetCDF data and visualization (matplotlib, netcdf4, cfgrib, cartopy, xarray).

The following table provides an overview of the data used or produced within this use case in terms of data format and usage.

Table 7. The format and usage of the data produced by the use case (section 5.3).

Data sets	**Data Flow**	**Persistency**	**Data type/ structure**	**Consumption pattern**	**Notes**
CMCC-CM3 input data	Sources	Persistent	NetCDF	1 to 1 (-> BB#1)	Grids, initial conditions, forcings, etc.
CMCC-CM3 output data	Intermediate	Persistent	NetCDF	N to 1 (BB#1 -> BB#2)	Data produced by the model simulation. It can consist of a large volume of data
CMIP6 Dataset	Sources	Persistent	NetCDF	N to 1 (-> BB#2)	Multiple models and variables required. It can consist of a large volume of data
IBTrACS	Sources	Persistent	Txt	1 to 1 (-> BB#3) 1 to 1 (-> BB#4)	Historical TC track observational data

ERA5 3-hourly reanalysis data	Sources	Persistent	NetCDF	1 to 1 (-> BB#3)	Required for ML model
Pre-processed CMCC-CM3 data	Intermediate	In-memory	ND arrays	1 to 1 (BB#2 -> BB#3) N to 1 (BB#2 -> BB#4)	Used for TC detection/tracking and multi-member/statistical analysis
Pre-processed CMIP6 data	Intermediate	In-memory	ND arrays	1 to 1 (BB#2 -> BB#3) N to 1 (BB#2 -> BB#4)	Used for TC detection/tracking and multi-member/statistical analysis
ML TC model	Intermediate	Persistent	HDF5	1 to 1 (-> BB#3)	Hyperparameters of the trained model (configuration, weights, etc.)
TC detection and tracking info	Intermediate/ Output	Persistent	NetCDF/ txt	N to 1 (BB#3 -> BB#4) (BB#3 ->)	NetCDF files with the TCs position or txt files with the sequence of TC positions
Statistical analysis results	Output	Persistent	NetCDF	(BB#4 ->)	NetCDF files with the results of the multi-member and statistical analysis
Maps with analysis results	Output	Persistent	PNG	(BB#3 ->) (BB#4 ->)	Visualisation of the various results obtained

5.4 General functional and non-functional requirements for Pillar II workflows

Taking into account information on building blocks and requirements of the ESM workflow and its components described above, in this section we provide the general functional and non-functional requirements. The keywords in the priority column are defined according to the RFC 2119 [11].

Table 8. General functional and non-functional requirements.

ID	Name	Description	Priority
1	Execution Robustness	Management of fault tolerance during the workflow execution including checkpoints or retries. For example, during a large execution if a node fails, the workflow must be able to recover and continue to the end.	Should
2	Portability	Workflow components should be portable to various types of HPC infrastructures.	Should
3	Integrated workflow management	Requires the Management of task dependencies, execution of parallel simulations on different HPC infrastructures, management of batch jobs (submission, monitoring, cancellation), management of conditional paths in a transparent way.	Must
4	Integration with long-term archive/repository storage	Results may be stored in long-term storage for archiving purposes, second use (e.g. downstream services) and/or to satisfy FAIRness policies.	May
5	Workflow adaptability	Capability to easily manage, cancel, replace and add components invocations in the workflow, for instance allowing the execution starting from the n-th step.	Should
6	Access to intermediate in-memory results	The workflow should be able to retrieve data/intermediate outputs of the running processes directly from memory.	Must
7	AI integration for ensemble member pruning	Support for applying Machine Learning techniques on intermediate data of running members to compute the members that will be discarded at a given step of the simulation.	Should
8	ML/DL capabilities	Requires the support for training and inference of Neural Network models for example for Tropical Cyclone detection.	Must
9	DA capabilities	Support for descriptive analytics (e.g., statistical analysis) exploiting fast in-memory analysis.	Must
10	High Performance Computing support	Climate models have to be executed on computing infrastructures capable of providing a large amount of processing and memory resources.	Must
11	Multi-member	Support for concurrent execution of sub-workflows starting from different inputs	Must

	analysis	(configurable) and comparison of the sub-workflows results.	

Besides the identified functional and non-functional requirements, it would be important to also ensure the following best practices that cover aspects related to code quality and development.

Test coverage: ESM models are multicomponent systems that have multiple dependencies in compilation and runtime. Changes in the system configuration or in the model code can introduce errors that are hard to find without continuous testing of the whole system (integration tests). Preferably continuous integration testing is implemented at the earliest stages of the development.

Inline documentation: This is mainly important for workflow developers, since it helps to understand the code and effectively implement their solutions.

Deployment and user documentation: The extent and quality of deployment documentation ensure smooth installation of all components of the workflow on new HPC systems by system administrators, and faster solution for possible problems. User documentation is a necessary component for the successful adoption of the workflows by large numbers of users.

6. Metrics for the evaluation of the developed workflows

This section provides the definition of metrics quantifying, for example, accessibility, reliability, scalability, performance, energy efficiency, and cost related to the Pillar II workflow. After defining a set of potential metrics, the list of key metrics, selected for the evaluation of the workflow during the project, is provided at the end of this section.

6.1 Definition of metrics

In the following, we will list metrics relevant for Pillar II from the collection suggested by WP1, and add additional metrics that are specific for our workflows.

Development & Maintenance

The complexity of the code should be held to a minimum. Ideally, users should not deal directly with the configurations of the underlying software components (e.g., PyCOMPSs and TOSCA), but manage the workflow through configuration files. Reducing the code complexity for workflow developers is also beneficial, and will increase adoption of the system. The concrete metrics that are relevant in this respect are:

Lines of Code: It measures the number of lines of codes in the source files associated with the workflows.

Cyclomatic Complexity: This measures the cyclomatic complexity from the source files.

Duplications: Measures the number of times the same code is repeated in the source files.

Decrease in all these three metrics will be considered an improvement.

Accessibility & Deployment

Users of ESMs often should be able to run different configurations of models on different machines. Even if the user always does a simulation in one HPC center, machines are changed every 5 years on average. The availability of the same software stack that manages workflow in different HPC centers is a key element for wide adoption of the eFlows4HPC solutions, which should be highly portable and easy to deploy.

Degree of portability: The landscape of ESM computations is changing, moving from homogeneous CPU-only systems to heterogeneous systems that include several architectures, like GPU and ARM. Moreover, novel storage solutions (e.g. NVRAM) are also becoming popular in HPC. Ability to use the **same or similar components** of the workflow on systems with different architectures, as well as on different infrastructures, is an advantageous quality of a workflow solution. The metric evaluates the percentage of workflow components capable of working on different infrastructures and with different architectures.

Deployment time: This measures the time elapsed to deploy the workflow.

Improvement in both of these metrics will foster adoption of the workflow on different HPC systems.

Data Management

Typically, ESM workflows produce large amounts of data that should be accessible for further analysis and also partly archived in long-term storage. One of the consequences of implementation of the dynamic part of the ESM workflow should be a reduction in the volume of data generated by ensemble simulations due to reduction in the number of simulations (caused by pruning), and due to in-memory analysis of data, that ease requirements on frequency of the model output.

Size of input/output data: For Pillar II, the analysis and feature extraction component of the workflow will try to increase the volume of data it can work with simultaneously (input data), while success for the dynamical component of the workflow will be in decreasing the amount of output data. This metric measures the amount of input and output data processed/generated.

IO time: The ESM workflows are intensive in terms of IO operations and reduction in the time of these operations will be beneficial. For the analysis and feature extraction component it would be the time data is loaded to memory for further analysis, while for the dynamical component of the workflow improvement in IO time should be achieved by decreasing the number of output fields and their frequency. It is measured as the percentage of execution time associated with I/O operations.

Reliability

Running ESM can be associated with a considerable number of failures that are related to complexity of the workflows. The failures can be caused by model physics (instability of the model in some regimes), but also the model's code, pre/post processing software, or problems with the infrastructure the ESM workflow is running on. In most cases re-running the whole workflow/simulation, which can last for many days, is unreasonable. All ESM models have mechanisms of checkpointing, so the simulation can be continued from some saved model state, while the other parts of the ESM workflow usually do not have mechanisms of recovery from the fault states. Implementing the ways that allow ESM simulations and post-processing jobs to recover from failures in the cases that can be fixed without necessary user interaction, would be beneficial.

Fault-tolerant components: In ESM workflows the simplest strategy on the failure is to retry execution of the part of the workflow, since the errors are quite often related to unpredictable failures in the infrastructure the workflow is executed on. For some parts of the workflow to recover from failures, checkpointing should be implemented; that will be considered an improvement in fault-tolerance. One has to keep in mind that checkpointing is usually an expensive operation in terms of I/O and it should be balanced in order to not degrade data management metrics excessively. Increase in the number of workflow components that can be retried on failure will be a positive improvement. This metric measures the percentage of components of the workflow that support some type of fault-tolerance mechanism.

Performance & Scalability

Performance is one of the most important characteristics to be considered for the success of the workflow. While both performance and scalability of the main components of the Pillar II workflows (i.e. ESM models) are out of the control of the workflow developers, the components of the workflow should not lead to considerable degradation of those parameters. Some decrease, though, is acceptable since it is not possible to have the additional benefits of, for example, a dynamical workflow, without additional computational cost.

For the feature extraction and analytics components, many of the tasks can be considered embarrassingly parallel (e.g. processing of data from individual CMIP6 models do not depend on each other) and should scale well.

Speed-up and Efficiency (Strong and weak scaling): Can only be improved for the components developed by eFlows4HPC, especially relating to data post-processing and analysis. Improvements will be measured by comparing strong and weak scaling before and after optimizations.

Execution time: The total workflow execution time should be as short as possible. This metric will exclude the computational components of the workflow, like model runs, but will include the components optimized by the project, as well as estimating the time spent on setting up the runtime of the workflow, preparing and submitting the simulation, logging and exiting from the workflow.

Energy Consumption

This metric is closely related to most of the Data management and Performance and Scalability metrics, where improvements could potentially lead to improvements also in energy consumption.

Energy consumption: This metric will be evaluated based on the estimated values of energy spent for all components of the workflow.

Cost

As the main resource the users of HPC systems are applying for is core hours, reduction in this parameter is the most attractive for ESM modelling groups, and will lead to better adoption of workflows developed by eFlows4HPC.

Core hours: This metric is closely related to improvements in the effectiveness of individual components of the workflow, as well as with more rational use of resources due to optimizations or reduction of computations necessary to reach specific scientific goals, as in case of the dynamical and AI assisted component of the Pillar II workflow. Decrease in this metric will be considered an improvement.

Others

Here we list a few additional metrics that are important mainly for Pillar II.

Accuracy of the results: The main criteria for this metric is that scientific questions that are addressed by the old methods should be addressable also by using novel approaches developed within eFlows4HPC. Results of the novel approaches should have comparable accuracy of the results. Optimisations in other metrics should not degrade the accuracy of scientific results. The evaluation of these metrics will be performed by comparing old and new approaches.

Simulated years per day: In ESM simulations increase in this metric is considered to be an improvement, and mostly controlled by the effectiveness of the ESM model. While improvements in the ESM model performance and scalability is not one of the tasks in eFlows4HPC, the dynamical component of the workflow potentially can reduce the I/O footprint in the ESM model execution and hence increase the SYPD values.

6.2 Final set of metrics selected

Starting from the metrics just presented a set of key metrics for Pillar II have been selected. The final list of selected metrics that will be used to evaluate the workflow implementation is:

Table 9. Key metrics for Pillar II.

Acronym	Name	Description	Area
LoC	Lines of Code	Number of Lines of code in the workflow implementation.	Development & Maintenance
DoP	Degree of Portability	Percentage of workflow components that can be reused in other infrastructures and workflows.	Accessibility & Deployment
DT	Deployment Time	Time elapsed to deploy the workflow.	Accessibility & Deployment
ET	Execution Time	Time elapsed to execute a workflow.	Performance
SU	Speed-up	Execution time improvement when running with larger resources.	Performance
Eff	Efficiency	Execution time degradation when running larger problems.	Performance
IOT	I/O Time	Percentage of Execution time performing I/O operations.	Data Management
FTC	Fault-tolerant components	Percentage of workflow components that are fault-tolerant .	Reliability
CH	Core/Hour	Number hours of a CPU Core consumed by the workflow execution.	Energy & Cost
EC	Energy Consumption	Energy consumed (Wh or Joules) associated with a workflow execution.	Energy & Cost

AR	Accuracy of the results	Accuracy of scientific results should not degrade.	Pillar II specific
SYPD	Simulated years per day	Throughput of ESM simulations.	Pillar II specific

7. Conclusion

The ESM modelling workflows of Pillar II consist of many building blocks of different complexity.

In this document we provide a short description of functionality associated with those building blocks, and draw a generalised set of requirements that software solutions should satisfy in order to execute them. For each building block, we define associated actions, input/output data structures, indication of HPDA/ML resource usages, deployment location, and typical time-scales. We also provide descriptions of data creation consumption patterns. In order to track the development and improvements in the eFlows4HPC solutions relevant to Pillar II, the set of evaluation criteria (metrics) are identified. We also provide descriptions on how Pillar II use cases will benefit from improvements in those metrics.

The requirements on the eFlow4HPC software stack from Pillar II defined in this document will guide the software architecture design that will be described in the D5.2 “Design of the Pillar II use cases”.

8. Acronyms and Abbreviations

Term or abbreviation	*Description*
CA	Consortium Agreement
CMIP	Coupled Model Intercomparison Project
D	Deliverable
DoA	Description of Action (Annex 1 of the Grant Agreement)
EB	Executive Board
EC	European Commision
ESM	Earth System Model
GA	General Assembly
GPU	Graphics Processing Unit
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
MPI	Message Passing Interface
MS	Milestones
PM	Person month / Project manager
TC	Tropical Cyclone
WP	Work Package
WPL	Work Package Leader
UC	Use case

9. References

[1] Villarini G., D.A. Lavers, E. Scoccimarro, M. Zhao, M.F. Wehner, G. Vecchi, T. Knutson, 2014: Sensitivity of Tropical Cyclone Rainfall to Idealized Global Scale Forcings Journal of Climate, doi: 10.1175/JCLI-D-13-00780.1

[2] Scoccimarro E., S. Gualdi, G. Villarini, G. Vecchi, M. Zhao, K. Walsh, A. Navarra, 2014: Intense precipitation events associated with landfalling tropical cyclones in response to a warmer climate and increased CO2. Journal of Climate, doi: 10.1175/JCLI-D-14-00065.1

[3] Scoccimarro E., P.G. Fogli. K. Reed, S. Gualdi, S.Masina, A. Navarra, 2017: Tropical cyclone interaction with the ocean: the role of high frequency (sub-daily) coupled processes. Journal of Climate , doi: 10.1175/JCLI-D-16-0292.1

[4] Scoccimarro E., S. Gualdi, A. Bellucci, A. Sanna , P.G. Fogli,E. Manzini, M. Vichi, P. Oddo, A. Navarra, 2011: Effects of Tropical Cyclones on Ocean Heat Transport in a High Resolution Coupled General Circulation Model. Journal of Climate, doi: 10.1175/2011JCLI4104.1

[5] Scoccimarro E., S. Gualdi, A. Navarra, 2012: Tropical Cyclone Effects on Arctic Sea Ice Variability. Geophysical Research Letters, 39, L17704, doi:10.1029/2012GL052987

[6] Horn M. ; K. Walsh; M. Zhao; S. Camargo; E. Scoccimarro; H. Murakami; H. Wang; A. Ballinger; A. Kumar; D. Shaevitz; J. Jonas; K. Oouchi, 2014: Tracking Scheme Dependence of Simulated Tropical Cyclone Response to Idealized Climate Simulations Journal of Climate , doi: 10.1175/JCLI-D-14-00200.1

[7] Cherchi, A., Fogli, P. G., Lovato, T., Peano, D., Iovino, D., Gualdi, S., et al. (2019). Global mean climate and main patterns of variability in the CMCC-CM2 coupled model. Journal of Advances in Modeling Earth Systems, 11. https://doi.org/10.1029/2018MS001369

[8] Scoccimarro E., Gualdi S., Bellucci A. , Peano D., Cherchi A., Vecchi G.A. , Navarra A. (2020): The typhoon-induced drying of the Maritime Continent. PNAS doi: 10.1073/pnas.1915364117.

[9] Eyring, V., Bony, S., Meehl, G. A., Senior, C. A., Stevens, B., Stouffer, R. J., & Taylor, K. E. (2016). Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization. Geoscientific Model Development, 9, 1937–1958. https://doi.org/10.5194/gmd-9-1937-2016

[10] Koldunov, N. V., Aizinger, V., Rakowsky, N., Scholz, P., Sidorenko, D., Danilov, S., and Jung, T.: Scalability and some optimization of the Finite-volumE Sea ice–Ocean Model, Version 2.0 (FESOM2), Geosci. Model Dev., 12, 3991–4012, https://doi.org/10.5194/gmd-12-3991-2019, 2019.

[11] Bradner, S. O. (1997). Key words for use in RFCs to Indicate Requirement Levels. RFC 2119. Request for Comments. https://rfc-editor.org/rfc/rfc2119.txt

10. Appendix

The following report the templates used for requirement identification for each block composing the workflow.

10.1 Requirements template from dynamical data analysis and AI assistance component

Building blocks Requirements: OpenIFS/FESOM2 simulation run (Use Case 1)

- Input/output data set
  - **Input:**
    - Initial conditions, forcing files, Netcdf or grib format, size: some GBs depending of the dataset, Datasets available on storage on execution time;
    - Mesh files for most common supported resolutions (Fesom2).
    - Model config files, outclasses files, usually on txt or XML format
  - **Output:**
    - netcdf files, size will depend on the resolution and mesh size, but at least several TBs are expected to be generated.
    - log and diagnostic files of the run, usually in format of txt files, for big experiments the size can be considerably bigger but less than 1 GB
- Computational Granularity:
    - Coarse grained (>secs)) since simulation can last several days
- Require specific software/hardware
  - **Software**:
    - OasisMCT,
    - MPI library (OpenMPI or Intel)
    - netCDF fortran library.
    - esm-tools (Python scripts) for running coupled system
    - CDO (climate data operators) for quick operation on netCDF files.
    - ncview for quick inspection of netCDF files
  - **Hardware**:
    - Enough infrastructure in number of cores and nodes to run desired model configurations
    - fast, reliable and parallel I/O, fast connection between cores and nodes
    - Enough storage to store intermediate and final results (> 1TB)
- Programming Languages
    - Bash
    - Python (numpy, pandas, matplotlib, netcdf4, cartopy, pyfesom2)
    - Fortran
    - C
    - C++

Building blocks Requirements: Pruning of the members (Use Case 2)

- Input/output data set
  - **Input**: set of intermediate restarts or set of model variables we consider important to evaluate to conduct the assessment for all the involved members of the ensemble
  - **Output**: an array of members that should be either discarded or continued depending of the criteria we will adopt on this (TBD)
- Computational Granularity: medium/fine-grain (<1sec)
  - The decision on which members will be discarded should be decided as fast as possible
- Require specific software/hardware
  - **Software:**
    - OpenIFS/FESOM2 model
    - netCDF fortran library.
    - esm-tools (Python scripts) for running coupled system
    - CDO (climate data operators) for quick operation on intermediate results, restarts (netCDF files) .
  - **Hardware**:
    - Ultra fast memory/ Non-volatile RAM memory if available
    - Enough storage to maintain the storage needs of the Hecuba as database/object repository
- Programming Languages
  - Bash
  - Python (numpy, pandas, matplotlib, netcdf4, cartopy, pyfesom2)
  - Fortran

Workflow deployment /execution requirements

- Locality
  - Initially intended to run on MN4 (MareNostrum4) but also it may be ported to AWI machine (Ollie)
- Licenses
  - Intel MPI
  - Intel compilers
  - FESOM2 (LGPL, free)
  - Hecuba (Apache License 2.0)
  - Open-IFS (ECMWF institutional license, BSC has it).
- Data availability:
  - initial conditions, forcing files, meshes and model sources are available in storage before start the execution of the model
- Expected execution requirements

- **ESM ensemble run (use case 1**): before running the model, all needed components are correctly deployed such as PyCOMPSS workflow engine, Hecuba and any other component or library needed to run an ensemble of the model
- **Member pruning (use case 2)**: before running, hecuba object repository is initialized, during execution, the pruning component to make the ESM dynamic can process the intermediate data or a subset of current model variable values at a defined step or after the execution of a given number of members

Data Requirements

- Data set:
    - Standard format for initial conditions and /forcing files: netcdf, grib
    - Small amount of initial conditions, periodically obtaining boundary conditions during the simulation, large amount of output data.
- Data flow: Sources / Intermediate data /Outputs:
    - **ESM ensemble run (use case 1**):
        - **Sources** : CMIP6 datasets; initial conditions in netcdf format
        - **Intermediate**: OpenIFS-Fesom2 model output/restarts, netcdf format
        - **Output**: netcdf files and text files with execution logs and diagnostics
    - **Member pruning (use case 2)**:
        - **Sources** : multidimensional arrays
        - **Intermediate**: N/A
        - **Output**: standard arrays/ numpy arrays
- Persistency requirements:
    - Disk as main use case, but for the dynamic workflow part some data should be persistent in memory
- Data types/structure (collections of small items, big items, arrays...):
    - Output is netCDF - large files, for the pruning component most likely to be numpy arrays
- Data creation-consumption pattern (1 to 1, 1 to N)
    - 1 to 1, Mostly data creation, factor depend on model resolution and frequency of output

10.2 Requirements template from analysis and feature extraction component

Building blocks Requirements: CMCC-CM3 simulation run

- Input/output data set

- **Input**: CMCC-CM3 initial conditions, radiative forcings (Netcdf format), size: few GBs; Datasets available on storage.
- **Output**: Netcdf: ~1 TB for 1 year of simulation

- Computational Granularity: coarse-grain(>secs)/ fine-grain (<1sec)
  - ~3h to simulate 1 month
- Require specific software/hardware
  - Model runs only on HPC at CMCC (necessary constraint for the duration of the project due to internal CMCC policies)
- Programming Languages
  - Fortran, bash scripts
- DA requirement
  - No DA operations required, only data movement and data organization processes (mv, cp, …)

Building blocks Requirements: Pre-processing

- Input/output data set
  - **Input**: CMIP6 or CMCC-CM3 datasets – Netcdf format
  - **Output**: Netcdf suitable for TC detection/tracking or Analytics blocks
- Computational Granularity: coarse-grain(>secs)/ fine-grain (<1sec)
  - It depends on the input datasets size; some seconds/minutes
- Require specific software/hardware
  - Netcdf-oriented tool: cdo (Climate data operators) or nco (NetCdf Operator)
- Programming Languages
  - bash script, python
- DA requirement
  - Concatenation of timesteps, regridding (if needed)
- ML requirements
  - None
- Integration of DA, ML with HPC kernels
  - Evaluate HPDA solutions to parallelize/speed up the execution

Building blocks Requirements: Feature extraction

- Input/output data set
  - **Input**
    - 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 from the pre-processing phase.
  - **Output**
    - NetCDF/txt with TC detection-tracking and statistical analysis results
- Computational Granularity: coarse-grain(>secs)/ fine-grain (<1sec)
  - ML NN Training: coarse grain
  - It depends on datasets input: some seconds/minutes

- Require specific software/hardware
    - **Hardware**: GPU (for NN training), CPU single core or MPI execution (depending on the specific implementation)
    - **Software**: specific tool for TC detection/tracking (eg. https://www.gfdl.noaa.gov/tstorms/), eflows4HPC ML tools/frameworks (e.g. Heat or EDDL), Python data science modules (netcdf4py, pandas, numpy), eflows4HPC data analytics tools/frameworks (e.g. Ophidia)
- Programming Languages
    - Fortran, Python
- DA requirement
    - Potential storm identification and storm tracking by means of specific algorithms of TC detection and tracking, computation of different statistical features (e.g. Nr of TCs per basin, distribution in the different categories, etc.)
- ML requirements
    - CNN for TC detection
- Integration of DA, ML with HPC kernels
    - Evaluate HPDA solutions to parallelize/speed up the execution (e.g. Ophidia)

Building blocks Requirements: Multimember/Statistical Analysis

- Input/output data set
    - **Input**: Pre-processed CMCC-CM3 dataset (Netcdf format), statistical analysis from feature extraction (Netcdf format), Observational best track data
    - **Output**: Netcdf (or txt) with indices/analytics results data
- Computational Granularity: coarse-grain(>secs)/ fine-grain (<1sec)
    - Few seconds/minutes
- Require specific software/hardware
    - numpy, pandas, matplotlib, netcdf4, cartopy, Ophidia, Netcdf-oriented software
- Programming Languages
    - Python, C
- DA requirement
    - Percentile/threshold based extreme events indices computation on temperature/precipitation (e.g. heat waves, …), multi-model trend analysis, multi-model intercomparison
- ML requirements
    - None
- Integration of DA, ML with HPC kernels
    - HPDA solutions to parallelize/speed up the execution (e.g. Ophidia)

Workflow deployment /execution requirements (for all blocks)

- Deployment restrictions
    - **Locality**: CMCC-CM3 model needs to run on CMCC cluster
    - **Licenses**: most of the software/code should be open source with the exception of the climate model simulation code and related output

  - **Data availability**: data from the model(s) is required for the execution of the pre-processing step, ERA5 data for training the ML TC detection, IBTrACS for TC detection/tracking validation and TC model comparison
- Expected execution requirements
  - **Use case 1**:
    - The workflow will also start the execution of the CMCC-CM3 model. The ESM simulation runs in parallel with the Feature extraction phase. For each iteration of the model execution, the aforementioned steps are executed at model runtime. Input data to the model must also be available on the storage.
    - ERA5 and IBTrACS should be accessible.
    - A further analytics block computes extreme events indices directly on model output.
  - **Use case 2**:
    - CMIP6 data should be available on storage.
    - A further block performs the multi-model analysis
  - **Common**:
    - The ML model should be pre-trained. Training should be executed offline on GPUs.
    - The pre-processing phase prepares the input for the subsequent feature extraction (TC detection and tracking and statistical analysis) phase along with observational data
    - The software components in the various stages (preprocessing, TC detection and tracking, statistical analysis) can be executed when required during the workflow. The Ophidia service components can be deployed in advance, while the computing components are executed on-demand.

Data Requirements (for all blocks)

- Data flow: Sources / Intermediate data /Outputs
  - **Sources**: CMIP6 datasets; CMCC-CM3 models initial conditions, radiative forcings, ERA5 reanalysis data, observational data (IBTrACS)
  - **Intermediate**: CMCC-CM3 model output, netcdf format at different time frequencies. 6hourly time frequencies files are input of the TC deterministic detection and tracking.
  - **Output**: netcdf or txt with information on TC detection & tracking and other statistical analysis (to be further evaluated).
- Persistency requirements (in –memory/disk)
  - Disk for the model output and the final results from TC track and detection
  - Other intermediate results could be in-memory
- Data types/structure (collections of small items, big items, arrays…)
  - NetCDF files: n-dimensional arrays (2, 3 or 4 dimensional)
  - text files with the TC positions (lat, lon, id)
- Data creation-consumption pattern (1 to 1, 1 to N) (stream…)

- Use case 1: N to 1 (Model simulation -> preprocessing); 1 to 1 (preprocessing -> TC detection & tracking & analysis)
- Use case 2: N to 1 (CMIP6 -> preprocessing); 1 to 1 (preprocessing -> TC detection & tracking & stats. analysis); M to 1 (stat. analysis -> multi-model analysis)