Enabling Dynamic and Intelligent Workflows for HPC, Big Data, and AI Convergence

Jorge Ejarque[a], Rosa M. Badia[a], Giovanni Aloisio[h], Enrico Baglione[k], Yolanda Becerra[a,c], Stefan Boschert[o], Alessandro D'Anca[h], Donatello Elia[h], François Exertier[f], Sandro Fiore[i], José Flich[e], Arnau Folch[m,a], Steven J. Gibbons[p], Md Sabbir Hasan[f], Nikolay Koldunov[l], Francesc Lordan[a], Stefano Lorito[k], Finn Løvholt[p], Jorge Macías[j], Fabrizio Marozzo[g], Alberto Michelini[k], Marisol Monterrubio-Velasco[a], Marta Pienkowska[n], Josep de la Puente[a], Anna Queralt[a], Enrique S. Quintana-Ortí[e], Juan E. Rodríguez[a], Fabrizio Romano[k], Riccardo Rossi[b,c], Jedrzej Rybicki[d], Jacopo Selva[k], Domenico Talia[g], Roberto Tonini[k], Paolo Trunfio[g], Manuela Volpe[k]

[a] *Barcelona Supercomputing Center (BSC)*
[b] *Centre Internacional de Mètodes Numèrics a l'Enginyeria (CIMNE)*
[c] *Universitat Politècnica de Catalunya (UPC)*
[d] *Jülich Supercomputing Centre (JSC)*
[e] *Universitat Politècnica de València (UPV)*
[f] *Atos BDS R&D HPC & AI Software*
[g] *DtoK Lab Srl*
[h] *Centro Euro-Mediterraneo sui Cambiamenti Climatici (CMCC)*
[i] *Department of Information Engineering and Computer Science, University of Trento*
[j] *Universidad de Málaga (UMA)*
[k] *Istituto Nazionale di Geofisica e Vulcanologia (INGV)*
[l] *Alfred-Wegener-Institut Helmholtz-Zentrum für Polar- und Meeresforschung*
[m] *Consejo Superior Investigaciones Cientificas (CSIC)*
[n] *Eidgenössische Technische Hochschule (ETH) Zürich*
[o] *Siemens AG*
[p] *Norwegian Geotechnical Institute (NGI)*

Abstract

The evolution of High-Performance Computing (HPC) platforms enables the design and execution of progressively more complex and larger workflow applications in these systems. The complexity comes not only from the number of elements that compose a workflow but also from the type of computations performed. While traditional HPC workflows include simulations and modelling tasks, current needs require in addition data analytics (DA) and artificial intelligence (AI) tasks. However, the development of these workflows is hampered by the lack of proper programming models and environments that support the integration of HPC, DA, and AI. What is more, there is a lack of tools to deploy and execute the workflows in HPC systems easily. To go further in this direction, this paper analyses the context of HPC/DA/AI convergence and presents use cases where these complex workflows are required. Based on this analysis, the paper presents the challenges of delivering a new workflow platform to manage complex workflows. Finally, it proposes a developing approach for such workflow platforms addressing these challenges, in two directions: first, defining a software stack that provides the functionalities to manage these complex workflows; and second, proposing the HPC Workflow as a Service (HPCWaaS) paradigm, which leverages the software stack to facilitate the reusability of complex workflows in federated HPC infrastructures. Proposals presented here are under design and development in the EuroHPC eFlows4HPC project.

Keywords: High Performance Computing, Distributed Computing, Parallel Programming, HPC-DA-AI Convergence, Workflow Development, Workflow Orchestration

1. Introduction

The scientific process has been described as consisting of three inference steps: abduction (i.e., guessing at an explanation), deduction (i.e., determining the necessary consequences of a set of propositions), and induction (i.e., making a sampling-based generalisation). These key logical elements have been presented in [1] by the Big Data and Extreme-Scale Computing (BDEC) [2], an international initiative that focuses on the convergence of data analytics (DA) and High-Performance Computing (HPC). While the abduction and induction phases imply the use of analysis and analytics processes (DA techniques), the deduction phase is typically an HPC process. However, the three different steps of the scientific process have been realised until now with separated methodologies and tools, with a lack of integration and a lack of common view of the whole process. The main BDEC recommendation is to address the basic problem of the split between the two paradigms: the HPC and Big Data software ecosystem split. In addition, current international roadmaps, including the BDEC, focus on combining HPC with artificial intelligence (AI), itself tightly linked to the big data revolution. Another observation is that the usage of HPC resources by scientific workflows is often done in a brute force manner where a large number of simulations or modelling

jobs are submitted, generating themselves a large amount of data which are later analysed/processed in a decoupled process. There is a need for smarter workflow approaches, able to use HPC in a more energy-efficient way but also able to perform the different HPC/DA/AI steps in a more integrated form. The situation is kind of similar in the context of industrial applications: for example, in the area of manufacturing; current technologies based on Full Order Models (FOM), developed for increasingly complex designs, generate a large amount of data that is processed in later steps to obtain Reduced Order Models (ROM) that can be used in the construction of digital twins. A more integrated approach will streamline the solution of FOM problems opening the door to adaptive algorithms. This, in turn, will allow faster and more reliable ROM, reducing the required simulation time thus improving the impact in the industry.

However, creating these new integrated workflows is not an easy task. Every HPC, DA or AI step of these workflows are implemented in a stand-alone framework designed for one purpose. So, developers have to dedicate a lot of effort to manage the integration of different frameworks in different phases of the workflow lifecycle. Starting from the development phase, where developers have to program the integration of the different workflow parts implemented in different programming models, passing through the deployment phase, where different tools and frameworks must be deployed in the infrastructure, and the execution phase, where the execution of all the different components must be orchestrated dynamically and intelligently way. For these reasons, new workflow platforms enabling the design of complex applications that integrate HPC processes, data analytics, and AI are necessary. These platforms should exploit the use of the HPC resources in an easy, efficient, and responsible way and also enable the accessibility and reusability of applications to reduce the time to solution.

To go further in this direction, this paper analyses the context of HPC/DA/AI convergence and presents use cases where these complex workflows are required. Based on this analysis, the paper presents the challenges of delivering a new workflow platform to manage complex workflows. Finally, it proposes a developing approach for such workflow platforms addressing these challenges. This platform consists of two parts: a software stack that provides the functionalities to manage these complex workflows, and the HPC Workflow as a Service (HPCWaaS) concept, which leverages the software stack to facilitate the reusability of complex workflows in federated HPC infrastructures[1].

The paper is organized as follows. Section 2 analyses the context of HPC/DA/AI convergence from different perspectives (development, deployment, data management and computer architecture). Section 3 presents use cases where complex workflows integrating different HPC/DA/AI techniques are required to efficiently solve different scientific and industrial problems. Section 4 presents the main challenges in efficiently supporting these new complex workflows, and Section 5 presents an approach to address these challenges. Finally, Section 6 provides an overview of related work and Section 7 draws conclusions and proposes guidelines for future research directions.

[1] In this paper, a federation refers to a set of HPC resources geographically distributed used in collaboration for a workflow execution

2. HPC, Big Data, and AI convergence

The recent wide availability of Big Data sources catalyzed a data-centric science based on the intelligent analysis of large data collections and on learning techniques for gleaning the rules hidden in them. Such data collections may consist of output from large HPC simulations, raw data from field and laboratory experiments, measurements of physical phenomena, from the Web, and in general produced in different scientific and engineering fields.

For implementing the integration of HPC/DA/AI solutions, new programming paradigms that enable the combination of HPC components with data analysis and AI algorithms must be designed.

As claimed by Dongarra and Reed [3], unification of HPC and Big Data analysis "is essential to address a spectrum of major research domains". To achieve this unification, adopted solutions must be general, portable, and extensible.

Scientific workflows offer researchers and developers a high-level and flexible programming paradigm for implementing a set of combined heterogeneous tasks expressing in a single application an entire scientific process or methodology that is too complex, expensive, or often impossible to implement in separate steps [4].

This section discusses the context and opportunities arising from the convergence of HPC, Big Data analysis, and AI solutions with specific attention to the development of a scientific workflow approach that may offer a high-level, smart, and practical paradigm for programming and running workflows that integrate HPC simulations, High-performance data analytics (HPDA), and scalable machine learning (ML) in a single application.

2.1. Workflow development perspective

We aim to define methodologies for the development of workflows where different aspects are integrated: data management and data analytics, high-performance computing, and AI processes. However, in current best practices, we find all these elements in separate components and environments.

2.1.1. General purpose workflows

Workflows provide a high-level paradigm for specifying the logic of an application, hiding the low-level details that are not fundamental for application design. They are also able to integrate existing software routines, data sets, and services in complex compositions that implement scientific discovery processes. The main issue in scientific workflow systems is the programming structures they provide to the developer who needs to implement a scientific application [4]. Various proposals have been formulated for the development of scientific workflows. Existing approaches in this area can be broadly categorized based on their level of abstraction (high-level versus low-level models) and on the type of programming formalism

they support; some are based on graphical interfaces, such as Kepler [5], Taverna [6] or Galaxy [7], some on textual interfaces, such as Pegasus [8], Askalon [9] or Autosubmit [10], and some on programmatic interfaces, such as COMPSs [11] or Swift [12].

Another aspect of the scientific workflows' environments is that each scientific community seems to stick to one or another solution. For example, Galaxy [13] has been adopted by the ELIXIR life science research infrastructure as its main workflow environment, while Cylc [14] was selected among others by the Earth Science community. An important component of workflow environments is their runtime or engine, which is responsible for coordinating the execution of all workflow tasks, scheduling them in the available computing resources, transferring the data between distributed storage systems, monitoring the execution of the tasks, etc. As with the interface, the runtime can be different from one environment to another, from very simple to more sophisticated ones, sometimes implementing different techniques to optimize the execution of the workflows and reduce the number of required data transfers, for example. Traditionally, workflow systems did not entail the possibility of supporting massively parallel or HPC tasks (tasks that run in parallel in large HPC infrastructures, implemented with MPI and/or OpenMP).

2.1.2. Development of HPC applications

Typically, HPC applications are developed using the Message Passing Interface (MPI) programming model [15], which is the de-facto standard for this type of application. It is based on the idea of having a large number of concurrent processes that exchange messages to solve a large problem cooperatively. Currently, MPI is combined with other approaches to exploit concurrency inside the large and fat HPC nodes. The most popular approach for this is OpenMP [16]. Additional complexity for the application developers is the appearance of accelerators such as the graphical processing units (GPUs) that require specific programming environments, like CUDA [17]. HPC programming models tend to be quite a low level and require a lot of effort from the application developer.

2.1.3. Data analysis workflows

Big Data analysis applications can be conveniently modelled as workflows combining distributed datasets, pre-processing tools, data mining and machine learning algorithms, and knowledge models. These workflows are able to integrate existing software modules, datasets, and services in complex compositions implementing discovery processes in scientific and business applications.

The compute and storage facilities of large-scale HPC systems can be effectively exploited to parallelize the execution of workflows composed of tens to thousands of tasks, to achieve higher throughput and to reduce turnaround times. This is particularly true in the context of Big Data analysis workflows, in which data volumes to be analyzed are huge, and tasks take a long time to complete their execution on conventional machines [18].

Implementing efficient Big Data analysis workflows from scratch on HPC systems is not trivial and requires expertise in parallel and distributed programming. For instance, it is necessary to express the task dependencies and their parallelism, to adopt mechanisms of synchronization and load balancing, and to properly manage memory and communication among tasks. In addition, when computing infrastructures are heterogeneous, different libraries and tools are required to interact with them. To cope with all these issues, high-performance programming models for data analysis workflows have recently being proposed [19].

As mentioned before, some key aspects of workflows are fostering the convergence between data analysis and HPC systems. Data analysis workflows allow programmers to express parallelism at different levels (i.e., data, task, pipeline parallelism), which can be exploited at runtime by HPC platforms composed of a large number of processing and storage elements. In addition, workflows can be designed to include several different patterns (e.g., cycles, filter, map-reduce, divide and conquer), whose variety helps programmers to address the needs of a wide range of applications and their mapping onto complex HPC platforms. Finally, the ability to reuse workflows by modifying the input data or the used algorithms and tools, combined with the ability to create hierarchical workflows where individual nodes can, in turn, be workflows, allow users to define and execute a variety of data analysis applications that goes well beyond the classical scientific applications executed on HPC platforms.

2.1.4. AI workflows

The convergence of AI environments, -and more specifically ML libraries-, with HPC platforms provide the opportunity for major performance improvement for the effectiveness of simulation, reusability, and reproducibility [20]. Usually, models are generated by running effective ML algorithms over large data sets that are produced from various sources. The generated model can comprise vectors of coefficients, different tree or graph structures with specific values. These derived models can accelerate the development of high-performance deep learning inference applications. Furthermore, pre-trained models speed up the production deployment process as well.

The foremost importance of having a model repository is to track the parameters and results of trained models to further package with ML libraries and codes in a reproducible and reusable manner in a targeted environment. For HPC and Big Data convergence, storing, managing, and sharing capabilities of models are key requirements for building workflows that make use of Machine Learning (ML) and Deep Learning (DL) models.

Furthermore, the deployment of these pre-trained models to a specifically targeted infrastructure or custom location by a scientific community may become a defined set of reusable workflow steps. Without the aforementioned capabilities surrounding models, users must undergo complex workflow steps at different levels: First, to generate models and second, to deploy them for usage in other workflow steps.

The key to achieving such ML/HPC convergence is to be able to exploit converged computing platforms for deployment of such workflows, where more specialized hardware resources (i.e., GPUs, FPGAs, etc.) are used to better support ML/DL workloads, together with HPC resources. Indeed, new scientific applications are made of the hybridization of traditional highly computational demanding simulations with ML/DL techniques. One may represent them as pipelines or even acyclic graphs (some parallelism is possible) chaining applicative modules having different kinds of execution paradigms. Therefore, the management of the life cycle of this kind of hybrid applications on top of heterogeneous resources is a challenge of the new generation orchestration stacks. Some related studies and experimentation have been conducted in the H2020 LEXIS project, both regarding the orchestration technology [21] and representative use cases [22].

2.2. Usage perspective

One of the main issues that HPC workflow developers and system administrators have to deal with is the installation and deployment of the workflow dependencies. Due to the widespread number of compilers, library versions, and their incompatibilities, every time users want to deploy a new workflow in a supercomputer, they have to check the installed dependencies, and install the missing ones taking into account the libraries and compiler versions to detect possible incompatibilities. To mitigate these issues, we can find some tools such as Spack [23] or Easybuild [24] which provide mechanisms to deal with these issues and automate the installation process of new software in HPC environments. However, they still require an expert HPC developer to create the packages or recipes for these tools and verify that they work for each supercomputer.

In Cloud environments, virtualization and container technologies have simplified the portability of complex applications. Hypervisors such as KVM [25] or container engines such as Docker [26] allow running processes in customized environments on top of computing nodes. These environments can be customized as normal computers and can be saved in images, which can easily be copied to other nodes where the same process can be executed with the same environment. The main barriers to applying these technologies in HPC are rooted in the requirement of running hypervisors and engines in privilege mode (root access) with the security consequences that this implies, and the integration of images with specific HPC hardware such as fast interconnects drivers. Singularity [27] is a container engine that tries to overcome these issues by not requiring privileged user mode to run the container and allows direct access to the host drivers to benefit from the special HPC hardware. Cloud Computing also provides service-oriented abstractions called Everything as a Service, where a set of services is offered depending on the usage model. One of the latest proposed service models is Function as a Service (FaaS). This service enables users to execute functions in the Cloud in a transparent way with a simple REST API call and without having to deal with the entire deployment, configuration and execution management overhead. The FaaS platforms, such as the commercial AWS Lambda, Google Cloud Functions, or the open-source approaches like OpenWhisk [28] or OpenFaaS [29], are in charge of managing the different function executions, allocating the computing resources when required, deploying the function software, getting the input data and storing the output results.

2.3. Data management/storage perspective

The huge amount of data involved in Big Data analysis workflows has a great impact on the overall performance of applications, as storage technologies have not kept up with the performance improvement in computing technologies, which are orders of magnitude faster.

Persistent storage in HPC has traditionally been dominated by file systems [30]. Applications consuming file-based data need to open and read the files and load data in memory, transforming it to the appropriate data structures for efficient manipulation. This process is usually implemented by the programmer as part of the application unless using specific file formats, such as NetCDF [31] or HDF5 [32], which provide specific libraries to facilitate this task. Additionally, scalability problems in file-based storage systems are well-known [30], which led to different kinds of storage solutions based on abstractions other than files (e.g. object stores or key-value stores, among others) gaining popularity not only in Cloud but also in HPC environments [33]. These storage solutions can provide more flexibility in accessing persistent data by enabling data accesses at a finer granularity, as well as providing efficient access to data during the computation, and facilitating the implementation of common application patterns in HPC, Big Data, and ML, such as producer-consumer or in-situ analysis or visualization. The challenge is to provide efficient storage solutions that allow the programmer to focus on the logic of the application and not be burdened by data access and data distribution details.

In addition, new technologies blurring the line between memory and storage have recently become available. These technologies, called persistent memories or non-volatile memories (NVM), such as Intel Optane DC [34], are similar to memory in speed, similar to disk in capacity, and are byte-addressable. These features open the door to computing directly on the stored data without having to bring it to memory, enabling HPC, Big Data, and ML applications to deal with larger volumes of data (i.e. not fitting in main memory) at high-speed [35]. The challenge is providing an easy way to exploit this technology that is transparent to the developer, letting the data store be in charge of managing the details of the interaction with the device in an efficient way.

2.4. Computer architecture perspective

With the end of Dennard's calling and the ever-increasing demands for higher performance from conventional HPC numerical codes, as well as emerging applications (such as, for instance, ML), in recent years the computer architecture scene has become much more diverse. We can identify processors that aim to augment the floating-point operations per second and Watt (FLOPS/W) following different paths such as very wide SIMD units (Fujitsu A64FX as ranked in the Green500

list [36]), very large count of cores (PEZY Computing's PEZY-SC2), using more conventional, yet heterogeneous designs combining general-purpose processors from ARM (Cortex-A), IBM (Power9) or Intel (Xeon) with NVIDIA's GPUs. The recent trends in HPC confirm that a hybrid architecture combining CPUs, GPUs, and even specialized accelerators has become the preferred node type for a large range of workloads of interest for HPC and data centres, including ML, Big Data, and scientific simulation.

The way to keep increasing performance while maintaining energy efficiency lies in the use of Domain-Specific Architectures (DSAs). Indeed, one of the most prominent and clear domains where specialization and adaptation of the system are appealing is Deep Learning. New accelerator units such as Google's TPU [37] offer considerable higher energy efficiency when compared with traditional architectures (CPUs or even GPUs). Adoption of such DSA architectures to workflows becomes significant. In addition, over the last years, reconfigurable devices, such as FPGAs, are gaining popularity as co-processing devices in HPC and Data centre environments. There are clear past successful examples such as the Catapult project [38]. Moreover, new products based on FPGAs such as Alveo boards or Versal boards target AI applications and the HPC domain. These devices also open the possibility to accurately balance the performance and energy efficiency objectives, for instance by adapting the precision arithmetic to the specific problem.

3. Use Cases

This section describes selected use cases from thematic areas with high industrial and social relevance, which can benefit from innovative and a more holistic workflow approach. These areas target very different users/communities and needs, and refer to: digital twins in manufacturing (Section 3.1), climate modelling (Section 3.2) and urgent computing for natural hazards (Section 3.3).

3.1. Digital twins in manufacturing

Today, the maturity of numerical methods allows the simulation of realistic problems in manufacturing and the definition of realistic digital counterparts, known as "Digital Twins" of the object or process of interest. Simulation-based design can nowadays largely substitute experimentation in many fields of application. The predictive value of the numerical models comes however at the price of a high computational cost. This becomes a blocker in different practical scenarios, and in particular in view of deploying the Digital Twin as a companion of the manufactured object for edge computing purposes (that is, for example on the on-board computer of production machines). This limitation can be solved by the use of Model Order Reduction approaches, which allow the definition of "surrogate models", known as "Reduced Order Models" (ROM) which present a similar predictive value although at a much reduced computational cost. The essential idea at the basis of such approaches is to perform first a campaign of high fidelity numerical experiments (known as Full Order Models or FOM) in order to collect training data. Such data is then analyzed in search of the most relevant patterns, typically by the use of large-scale Singular Value Decomposition (SVD) techniques. Finally, the identified patterns are fed back to the original simulation model which exploits them to construct the target ROM model. The corresponding workflow is shown in Figure 1.

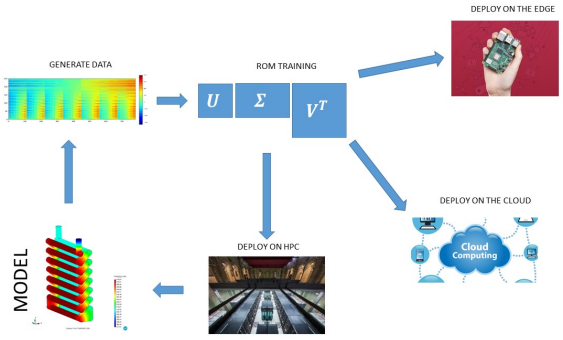


Figure 1: Main phases in the Pillar I workflow for the construction of ROM models.

The overall outcome is that the ROM model provides a tunable approximation (i.e. an approximation with a controllable level of accuracy with respect to the original FOM model) albeit at a fraction of the cost and of the memory required by the FOM. Projection-based ROM approaches such as the ones described can be understood as a special class of ML techniques, characterized by an overall workflow that follows the classical training and inference model. From the computational point of view, the training part is particularly challenging since it requires dealing both with the generation of "experimental" data and with their analysis via large scale SVD. The computation of the SVD is identified as the single computationally critical kernel in the workflow. The problem will be tackled by the use of a distributed Randomized Truncated SVD for which a task-oriented implementation will be developed.

A complete workflow may also require an iterative refinement of the training campaign to deal with gaps in the training space. The effective use of supercomputers requires integrating both the training and the inference steps within a single complex workflow to be adapted to the specific needs of the problem to be addressed. A practical challenge is related to the need to deploy the different software stacks involved on multiple hardware configurations. This represents a nontrivial challenge given the hard dependency on system libraries, such as MPI.

Furthermore, the described workflow can also be integrated with other AI frameworks with the aim of eventually employing the constructed ROMs as building blocks in the construction of system-level models. This integration also poses challenges, particularly regarding the interoperability between the different modules to be integrated within the same workflow.

3.2. Climate Modelling

The study of climate change and related climate phenomena is extremely challenging and requires access to very high-

resolution data. In this respect, the climate community has been continuously pushing the boundaries to implement and run model simulations at the highest resolution possible, exploiting cutting-edge supercomputing infrastructures. The resulting output consists of large, complex, and heterogeneous datasets which require proper solutions for management and knowledge extraction [39] and can take advantage of data-oriented approaches from DA and AI fields.

Typical end-to-end Earth System Modelling (ESM) workflows consist of different steps including input data pre-processing, numerical simulation runs, output data post-processing, as well as data analytics and visualization. Even though they represent different parts of the same scientific discovery process, their *seamless*, *intelligent*, and *efficient* integration into HPC environments still needs to be addressed at different levels to become a reality.

The methodologies currently available to develop scientific workflows in the climate field are, in fact, not able to seamlessly and transparently integrate the whole set of components (including DA and AI) into a single end-to-end ESM workflow. Thus, ESM workflows can greatly benefit of enhanced workflow solutions.

Figure 2 shows an innovative end-to-end ESM workflow which can be made possible through the integration of HPC, DA and AI components.

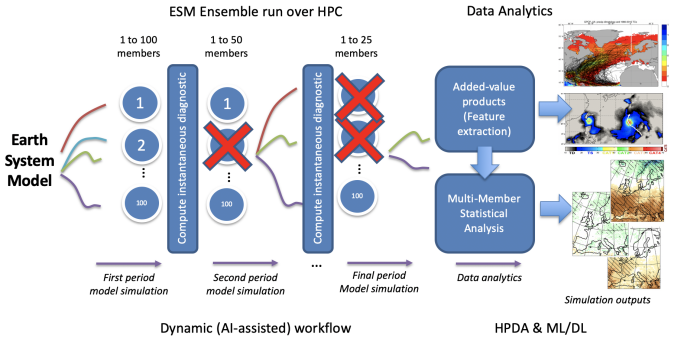


Figure 2: Main phases in the ESM workflow.

The advancements provided by dynamic access to the model simulation results at runtime, together with AI techniques can be exploited as part of the ESM workflow management. They can bring forward advanced possibilities for smart execution of the workflow, enabling more efficient resource usage as well as a shorter time-to-solution. One of the typical tasks in the climate modelling is to run ensemble simulations that consist of multiple members and can take a significant amount of time. Ensembles are used to improve our certainty in model results, for model tuning, or for exploring different scenarios of particular events. Usually, the number of members to run is limited by the amount of computational resources available, while not all the members might be needed at the end. In this sense, dynamic workflows with in-memory access to model results, able to adapt simulations at runtime by performing smart (possibly AI-driven) pruning of ensemble members, could thus reduce resource usage and impact on energy efficiency. One of our major objectives is to determine which metrics can be used to prune members without impacting the quality of the simulation. This also requires systems able to dynamically adapt the workflow execution based on mentioned runtime computed metrics. In a more general sense, dynamical access to model results allows implementation of model diagnostics, especially those that require high temporal frequency data without changing the model code and frequent data serialization. This is especially important for very high resolution climate models, that are challenged by I/O and storage limitations.

Data-driven approaches can also play a significant role in enhancing knowledge extraction from large climate simulation data, for example with respect to multi-model/multi-member ensemble and extreme events analysis, leading to a better understanding of the climate system. In this respect, Tropical Cyclone (TC) detection and tracking represents an important case study since it requires multiple two-dimensional fields, such as *pressure, temperature, wind velocity, vorticity*, at different time steps (with a frequency of at least 6 hours) and from very high-resolution General Circulation Models (GCM) data, for example coming from the Coupled Model Intercomparison Project - phase 6 (CMIP6) [40] or very high-resolution models (e.g. the CMCC-CM3 model).

TC detection and tracking analysis can hence be very challenging due to the large amount of data involved, its heterogeneity, and processing complexity. This is even more critical if data from multiple models are considered in the process. Different detection and tracking methods are available in the literature [41] and new emerging approaches are investigating the use of ML/DL techniques to verify the possibility to speed-up the process and make it more energy efficient in the context of a multi-model multi-member analysis.

These types of analyses will hence greatly benefit from the inclusion of the DA and ML/DL technologies in the HPC workflow. The adoption of more integrated and data-driven approaches will enable scientists to tackle much larger and more complex science problems than are possible today in the climate change domain. In-situ mechanisms will represent another step forward in this direction by integrating data-driven approaches directly within the model simulation, thus delivering an even more efficient solution. The outcome of these analyses can represent added-value datasets provided to end users to support the development of new downstream services.

These enhanced workflow capabilities will ultimately (i) support transparent integration of simulation-centric and data-driven components, (ii) allow scientists to further increase knowledge of the climate system by delivering better data to end users for societal challenges and (iii) democratize access to these complex end-to-end ESM workflows.

3.3. Urgent Computing for Natural Hazards

In a general sense, urgent computing (UC) refers to the use of HPC/DA during or immediately after emergency situations and typically combines complex edge-to-end workflows with capacity computing, where multiple model realizations are required (to account for input and model uncertainties) under

strict time-to-solution constraints. Early decisions in rapid response to earthquakes have to be based upon interpretations of the best, yet often limited, data available at a given time immediately following the event to estimate the severity of shaking and, potentially, the impact of a subsequent tsunami. A combination of data analysis and numerical modelling, starting from a large ensemble of sources describing the uncertainty stemming from limited data, produces maps of strong ground shaking and/or tsunami inundation which can be employed to forecast losses (e.g. towards the insurance industry) and direct immediate relief measures (e.g. towards civil protection and first and second responders). The temporal horizon for seismic and tsunami urgent computing typically ranges from minutes to a few hours. For instance it is around 2 hours for the phenomena investigated in the ARISTOTLE-eENHSP (enhanced European Natural Hazard Scientific Partnership) project [42].

Numerical earthquake and tsunami simulations can accurately model the wave propagation, but computed outcomes are very sensitive to uncertainties in the source characteristics (e.g. earthquake type, size, structure, location, slip distribution) and boundary conditions (e.g. earthquake wave path, local site conditions), which may result in large variability of the impact estimate [43, 44]. Such input data sensitivity, exacerbated by limited data availability shortly after an event, have rendered large-scale ensemble numerical simulation a necessary analysis tool to study both past and hypothetical future earthquakes and their associated tsunamis and, considering their inherent computational cost, only HPC resources enable the use of high-fidelity geophysical simulations within the required time-to-solution constraints.

Seismic and tsunami urgent computing workflows consist of three main phases (Figure 3):

1. A pre-processing phase, in which an ensemble of possible sources is defined based on seismic data assimilation and earthquake parameter estimation, with uncertainty.
2. A simulation phase, in which individual scenarios are simulated and ground shaking / tsunami inundation numerically quantified for all the scenarios in the ensemble
3. A post-processing phase, in which simulation results are processed to produce probabilistic forecasts including both source and modelling uncertainty, eventually updated through newly available observations from the monitoring networks.

3.3.1. *Probabilistic Tsunami Forecasting and Faster Than Real Time tsunami simulations*

Probabilistic Tsunami Forecasting (PTF) for tsunami early-warning and urgent computing is based on the generation of an ensemble of Faster Than Real Time (FTRT) tsunami simulations, based on source estimate available immediately after the (potentially tsunamigenic) earthquake [45, 46, 47, 48, 49]. Complete workflows for PTF and FTRT have been developed as Pilot Demonstrators in the ChEESE Center of Excellence. Uncertainty arise from both the scarce knowledge of fault geometry and mechanism, and the limitations in tsunami modelling. Furthermore, PTF manages this with large ensemble of

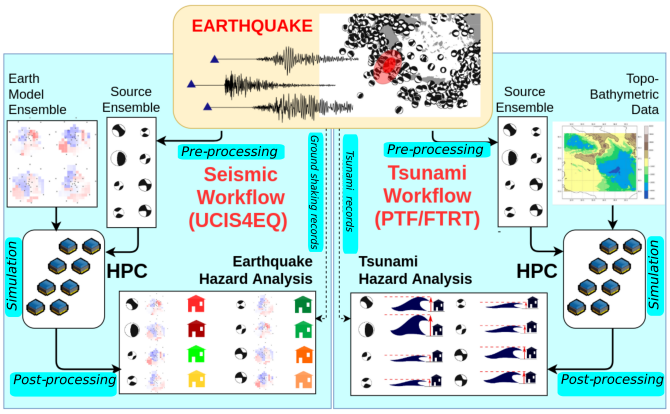


Figure 3: Main phases in the Urgent Computing workflows for Earthquakes and Tsunamis. See text for more details.

FTRT tsunami numerical simulations coupled with the management of the inherent modelling uncertainty. Within the current PTF workflow, the ensemble initialization is connected to standard seismic monitoring tools, whereas the subsequent FTRT tsunami simulations are carried out by the Tsunami-HySEA GPU-based software. The post-processing phase aggregates simulations, treating inherent uncertainty. In this phase, first order results are used for rapid post-processing, while detailed results are stored for subsequent Big Data detailed analyses.

Further aspects of the integration of PTF and FTRT to improve the operational level of tsunami forecasting are:

1. revision of the PTF workflow to reach performance time-to-solution targets suitable for urgent computing (~ 2 hr);
2. optimization of the ensemble initialization and updating based on real-time seismic source data, tsunami recording and DA, to produce dynamically evolving uncertainty quantification;
3. use of AI for rapid tsunami estimation (e.g. [50, 51, 52]) to enhance the FTRT workflow and potentially integrate ensembles in real-time;
4. use of DA and AI tools to enhance event diagnostics and in deeper post processing analyses (like scenario disaggregation).

3.3.2. *UCIS4EQ*

The Urgent Computing Integrated Services for Earthquakes (UCIS4EQ) workflow has been developed as a Pilot Demonstrator under the ChEESE Center of Excellence. UCIS4EQ aims at obtaining physically-consistent shaking estimates shortly after the occurrence of earthquakes by means of large 3D full waveform simulations [53]. UCIS4EQ is coupled to state-of-the-art and massively parallel simulation solvers so that, if large HPC resources are readily available, very fast simulations can be run in order to produce valuable outputs within minutes to hours. Typical uncertainties in such simulations come from source characteristics that cannot be constrained uniquely at the moment of issuing the forecast, and soil effects, which may (de)amplify seismic waves and cannot be accounted for in coarse

simulation models. UCIS4EQ includes capabilities to incorporate such uncertainties within the service. UCIS4EQ is designed and developed considering not only the functional requirements of the service, but also ensuring the quality of key non-functional requirements such as robustness, interoperability, availability and maintainability. To this purpose, each process is encapsulated to work as a specialised micro-service, the architectural building blocks of UCIS4EQ, with all components containerized.

Further aspects to be upgraded in the UCIS4EQ workflow to bring it closer to an operational level are:

1. The implementation of novel features addressing workflow monitoring and steering, including dynamic management of resources, extended output post-processing on demand or ensemble simulation capabilities.
2. Generating a database of 3D velocity models obtained from full-waveform inversions for a set of regions. Such models will be ready for urgent simulations and should support frequencies as high as currently possible given the data coverage in each region.
3. Integrating the whole UCIS4EQ workflow in an integrated workflow management system.
4. Adding regression or deep learning models to estimate intensity maps (PGA, PSA, among others) at time scales of seconds to minutes. Such ML analogs, given the fast outputs that they can produce, will allow us to explore uncertainty quantification. The ML models will be developed/trained based upon a large data-sets of physics-based earthquake simulations.

4. Challenges

The context and requirements of the described use cases have raised a set of challenges in workflow design and management that are summarised below.

4.1. Challenge 1: Enable the openness, transparency, reusability, reproducibility and accessibility of the workflows and their results

With the introduction of virtualization and containers, porting applications into Cloud environments has improved considerably, facilitating easy porting of applications implemented in well-known software stacks (e.g. LAMPS, MEAN, Hadoop). However, the e-science workflows' complexity is increasing fast, and they are required to combine multiple HPC, Big Data and AI frameworks. Enabling the portability and accessibility of these workflows on the wide variety of HPC systems is still an open challenge. First, because current e-science workflows require the deployment and orchestration of several frameworks, which must be coupled tightly to the computing infrastructure to benefit from the HPC infrastructure. Moreover, reusing the same tools used in the Cloud environments to deploy applications is not possible in most HPC systems due to the security and accessibility requirements in supercomputers. Therefore, installations and deployments are usually managed by system administrators to ensure they are adapted to the supercomputer capabilities. A similar challenge appears with the workflow results. Enabling their reusability and reproducibility requires designing and implementing a tailored mechanism to make results available to the users considering the access restrictions of the HPC systems. To overcome this situation, new tools or current Cloud deployment and data-sharing tools have to be redesigned, extended, and adapted to accommodate the requirements of the new complex workflows and to fulfil the HPC access constraints.

4.2. Challenge 2: Simplify the development of complex workflows while keeping their capabilities and performance

Traditionally, the HPC software stack has focused on providing libraries to optimally exploit the available infrastructure. Existing HPC programming models such as MPI or OpenMP enable the development of parallel applications but they are still too complex for general scientists, especially if they need to develop a higher abstraction, complex, workflow.

Also, as seen in section 2, different methodologies have been proposed for the development of HPC codes, DA, and ML. In general, current methodologies available to develop scientific workflows do not fulfil the requirements of increasingly complex applications, which require novel methodologies supporting a holistic workflow composed of HPC simulations or modelling, data analytics, and machine learning.

To keep performance, the usage of new, powerful, and energy-efficient heterogeneous computing nodes is a must. For example, GPUs are not only extensively used in traditional HPC codes but are also very efficient in the DL training phase. However, the complexity of programming heterogeneous devices has been recognized by the community and multiple efforts to overcome this challenge have been proposed [54, 55, 56].

To address these issues, new methodologies that support simpler and intuitive workflow development need to be proposed: focusing on workflows that integrate components of diverse nature (HPC, AI, and DA). These methodologies should be able to bridge the gap between the application and the heterogeneous infrastructure in order to maintain the expected performance.

4.3. Challenge 3: Support for workflow dynamicity

Another challenge introduced by complex workflows is dynamicity. Current workflow managers support static workflows whose dynamicity is very limited. This approach fits with traditional workflows that solve problems with simple pipelines or graphs, repeated loops with different input parameters, or small workflow modifications using conditional control flows. However, as stated in Section 3, the new complex workflows considered in this paper require support for high degrees of dynamicity and flexibility in the workflow development and execution. The workflow programming models and engines have to be able to support applications with dynamic data sources with input values constantly changing and can produce alterations in the computation workflow, invalidating the initiated computations and requesting new computations. Moreover, this adaptation must be applied in real-time to fulfil the urgency constraint

of the computation. To support this dynamicity, the workflow manager should be able to react to changes in the input data and generate new computations on demand.

However, the workflow dynamicity is often driven by more than the input data. In some cases, an early analysis of the results can detect parts of the workflows tending to solutions that are insignificant for the final results, such that these computations can be cancelled for saving resources or dedicating them to extend the search space or increasing the effectiveness of solutions. In this sense, the workflow manager should be also able to modify an existing workflow by removing already expected computations, and sometimes adding new ones in reaction to given events.

Finally, the mentioned dynamic workflow support at the development and execution level has to be tightly combined with elastic resource management in order to adapt the computing capacity with the changing computing demands required by the workflows, which will make more sensible use of the resources and save power.

4.4. Challenge 4: Enable data management and computation integration

Current approaches to implementing data management usually differ between environments that execute scientific applications and those that implement data analytics. However, workflows integrating both kinds of computation can improve the productivity of scientists, as well as the performance of workflows themselves, which could start the analysis of the data before the data generation ends. To provide an effective environment for this kind of hybrid workflows it is necessary to provide a unique data management strategy that reduces the data movement between different storage systems, whilst supporting both scientific and data analytics workloads efficiently. This unique strategy should be able to provide the data generation application with a fast data ingestion mechanism, that should not limit the potential parallelism of the computation (avoiding synchronisation points due to data storing). And at the same time, it should be able to provide a simple interface that enables programmers to access partial results efficiently.

Datastores for data analytics usually meet these requirements. However, developers of HPC applications, used to working with files, are reluctant to adopt them for several reasons. First, the efficient utilisation of this type of datastores involves a low-level knowledge of their design, to tune all the available configuration parameters. Second, deciding how to organize the data (i.e. defining the data model) influences the performance of both reading and writing, and getting efficient data models also requires a deep knowledge of the execution platform (both hardware and software stack). Third, enhancing data locality is also a goal of this type of data store, but once again usually it is not transparent to programmers. Finally, changing their traditional approach to storing data involves learning new interfaces that usually change from one datastore to another.

To overcome this reluctance, it is necessary to add a layer to the software stack that relieves programmers of these tasks. This layer should provide automatic and transparent tuning of the data store and data locality enhancement, automatic data modelling, and a simple interface independent of the particular data store in the system and close to the data structures managed by the application. These features would allow the programmer to focus on the problem domain, and at the same time, provide the required performance and parallelism in collaboration with the programming model.

In addition, the popularisation of persistent memory devices offers the possibility to rethink strategies both on how data is accessed and how data is modelled in datastores. By exploiting the capabilities of these devices in the data management layer, again transparently to the programmer, applications will be able to seamlessly manage larger amounts of data and benefit from a higher performance in data access.

5. eFlows4HPC solution

eFlows4HPC is a EuroHPC funded project which aims at enabling dynamic and intelligent workflows in the future European HPC ecosystem. High-level projects structure is depicted in Figure 4. We propose integrated solutions to cover the challenges presented in Section 4. First, eFlows4HPC defines a software stack that covers the different functionalities to support the whole lifecycle of the complex workflows introduced in this paper. Second, it proposes the HPC Workflow as a Service (HPCWaaS) methodology to enable reusability of these complex workflows as well as simplifying the accessibility to HPC resources. Finally, the project also works on the workflow kernels for new heterogeneous architectures.

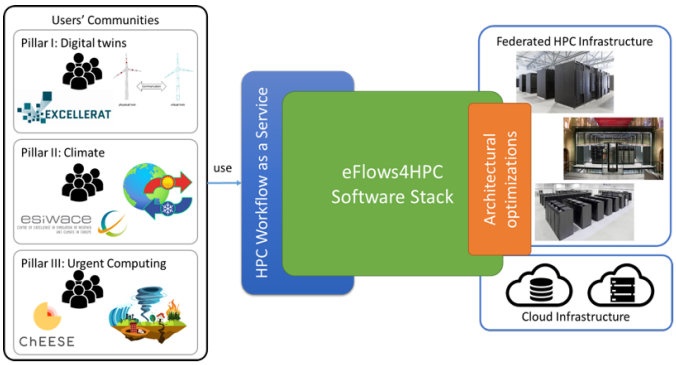


Figure 4: eFlows4HPC project overall approach.

5.1. eFlows4HPC software stack

The eFlows4HPC software stack will comprise existing software components, integrated and organised in different layers (Figure 5). The first layer consists of a set of services, repositories, catalogues, and registries to facilitate the accessibility and re-usability of the implemented workflows (Workflow Registry), their core software components such as HP libraries and DA/AI frameworks (Software Catalog), and its data sources and results such as ML models (Data Catalog and Model Repository). The second layer provides the syntax and programming models to implement these complex workflows combining HPC simulations with DA and AI. A workflow implementation consists of three main parts: a description about how the software

components are deployed in the infrastructure (provided by an extended TOSCA definition [57]); the functional programming of the parallel workflow (provided by the PyCOMPSs programming model [58]); and data logistic pipelines to describe data movement to ensure the workflow data are available in the computing infrastructure when required. Finally, the lowest layers provide the functionalities to deploy and execute the workflow based on the provided workflow description. From one side, this layer provides the components to orchestrate the deployment and coordinated execution of the workflow components in federated computing infrastructures. On the data management side, it provides a set of components to manage and simplify the integration of large volumes of data from different sources and locations with the workflow execution.

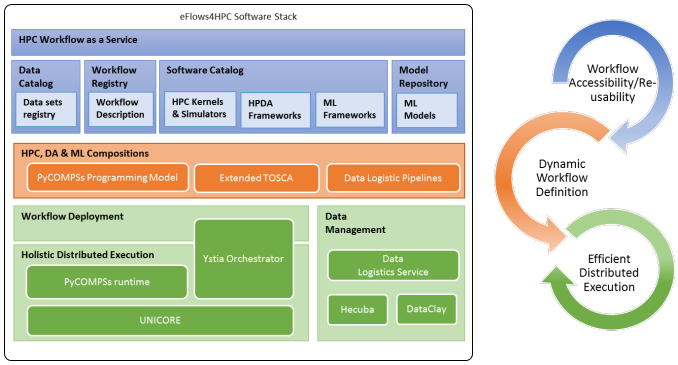


Figure 5: eFlows4HPC Software Stack.

5.2. *HPC Workflows as a Service (HPCWaaS)*

Currently, one of the main barriers to the adoption of HPC is the complexity of deploying and executing workflows in federated HPC environments. Usually, users are required to perform software installations in complex systems which are beyond their technical skills. Therefore, having the workflows ready for execution in a supercomputer could take large amounts of time and human resources. If it needs to be replicated for reliability requirements to several clusters, the required time and resources will increase. To widen the access to HPC to newcomers, and, in general, to simplify the deployment and execution of complex workflows in HPC systems, eFlows4HPC proposes a mechanism to offer HPC Workflows as a Service (HPCWaaS) following a similar concept as the Function as a Service (FaaS) in the Cloud, but applying it for workflows in federated HPC environments. The goal is to hide all the HPC deployment and execution complexity to end users in such a way that executing a workflow only requires a simple REST web-service [59] call. It will also provide a mechanism to enable the sharing, reuse, and reproducibility of complex workflows.

Figure 6 shows an overview of how the proposed model works. The HPCWaaS is built on top of the eFlows4HPC software stack in order to provide the required functionality to develop, deploy, and execute complex services. The interaction of the users with HPCWaaS is done in two phases: one for workflow developers and another for workflow user communities. At

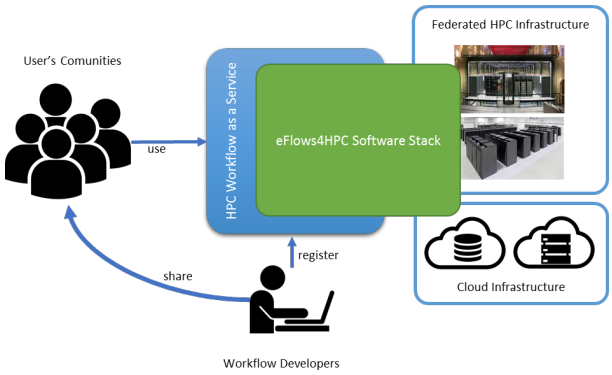


Figure 6: Overview of the proposed HPC Workflow as a Service model.

development time, workflow developers are in charge of building the workflow using the first two layers of the eFlows4HPC stack. Once the workflow implementation is compleated, the workflow is registered in the HPCWaaS to make it available to the whole community of the target sector. After successful registration, the workflow developer receives a service endpoint from the HPCWaaS, that other users can invoke to use the developed workflow which will be automatically deployed and executed in the computing infrastructure using the rest of eFlows4HPC stack functionalities.

The following paragraphs provides more details about how the different eFlows4HPC components interact to provide the required functionality in the different phases.

5.2.1. *Workflow development phase*

One key part of the mentioned challenges is the implementation of complex workflows that combine HPC, DA, and AI frameworks. eFlows4HPC proposes two mechanisms in order to achieve this challenge as depicted in Figure 7. On the one hand, the software stack provides a set of registries, catalogs and repositories, providing workflow developers with the means to store the core components (HPC, DA, and AI frameworks) and the required data and ML models in such a way that they can be easily reused and combined. On the other hand, we propose the definition of a workflow description which enables the combination of the different workflow components. From this workflow description, the third layer of the eFlows4HPC software stack can be used to automatically deploy and execute the workflow in the Computing Infrastructures.

The proposed workflow description is composed of a combination of an Extended TOSCA syntax, the PyCOMPSs programming model and a set of data logistic pipelines. In the first part, TOSCA (an orchestration standard) allows developers to specify which software and services are required; and how each component should be deployed, configured (linked to each other), started, stopped and deleted. In the second part, the PyCOMPSs programming model will provide the logic of the different components of the overall workflow. PyCOMPSs is a task-based programming model that enables the development of workflows that are executed in parallel on distributed computing platforms. It is based on programming sequential Python scripts, offering the programmer the illusion of a sin-

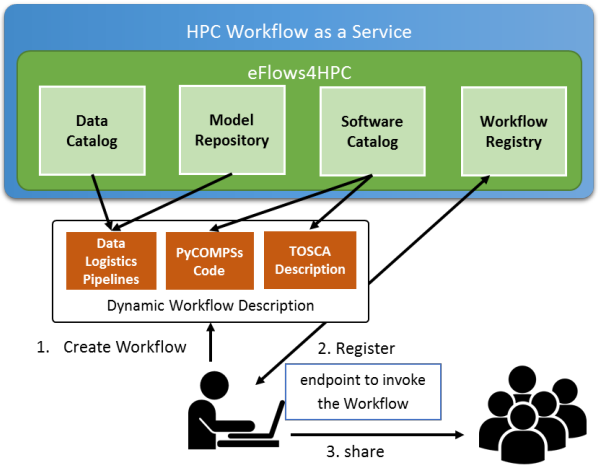


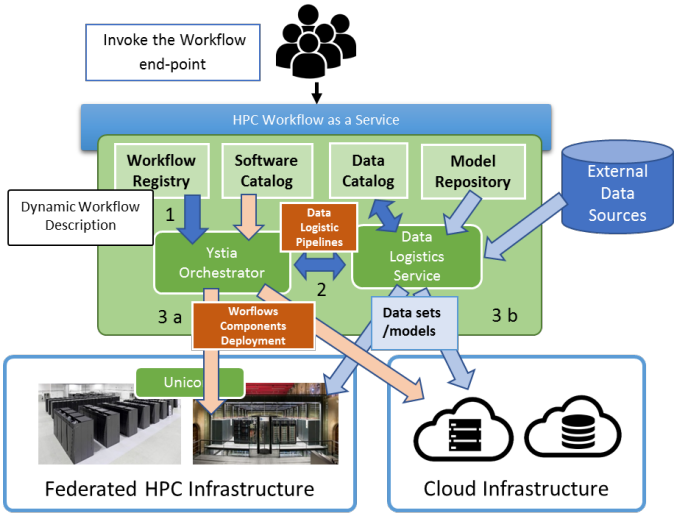
Figure 7: Workflow development phase.

gle shared memory and storage space. While the PyCOMPSs task-orchestration code needs to be written in Python, it supports different types of tasks, such as Python methods, external binaries, multi-threaded (internally parallelised with alternative programming models such as OpenMP or pthreads), or multi-node (MPI applications). Thanks to the use of Python as the programming language, PyCOMPSs naturally integrates well with data analytics and machine learning libraries, most of them offering a Python interface. Finally, in the last part of the workflow description, the data logistic pipelines allow developers to describe how the workflow data is acquired, moved and stored during the workflow execution in order to ensure the data is available in the computing infrastructure when required. The pipelines are also defined in Python, which reduces the entry barrier for the development.

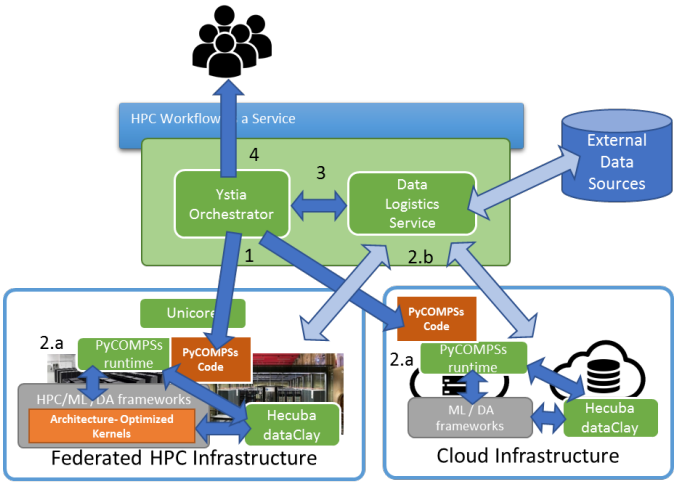
As mentioned before, the workflow description is registered and stored in a workflow registry by means of the HPCWaaS interface. The result of this registration will produce a service endpoint that can be later used to invoke the execution of the workflow.

5.2.2. *Workflow invocation phase*

When users want to execute the registered workflows, they only need to invoke the endpoint provided at the end of the workflow development phase. As a result of this invocation, the last layers of the eFlows4HPC software stack are used to provide an automatic and holistic workflow deployment and execution in federated computing HPC infrastructures. This functionality is provided, as depicted in Figure 8, by the cooperation of several components at different levels. At the highest level, the Ystia Orchestrator (Yorc) is in charge of managing the overall workflow deployment and execution. First, it retrieves the workflow description (Step 1 in Figure 8a) and passes the data logistic pipelines to the Data Logistic Service (Step 2 in Figure 8a) to set up the required data movements such as the data stage-in and stage-out, or periodical transfers to synchronize data produced outside the HPC systems (Step 3b in Figure 8a and Step 2b in Figure 8b). In parallel with the data deployment, Yorc orchestrates the deployment of the main workflow components in the computing infrastructures and managing their lifecycle (configuring, starting services) as described in the TOSCA part in the workflow description(Step 3a in Figure 8a).



(a) Workflow deployment.



(b) Workflow execution.

Figure 8: Workflows invocation phase. Dark blue arrows represent eFlows4HPC component interactions, light blue represent data flows, and orange arrows represent component deployments.

Once the workflow components and initial data are deployed, Yorc submits the execution of the main workflow processes in the HPC infrastructure through UNICORE [60], which is in charge of managing the federation of HPC compute and data resources in order to make them available to users in a secure way (Step 1 in Figure 8b). At the lowest level, the COMPSs runtime [61] will coordinate the invocations of the workflow components implemented with the PyCOMPSs task-based programming model (Step 2a in Figure 8b). As mentioned before, COMPSs supports several task types which can include HPC simulations, DA transformations, etc. The runtime dynamically generates a task-dependency graph by analysing the existing data dependencies between the invocations of tasks defined in the Python code. The task-graph encodes the existing

parallelism of the workflow, which can be used to schedule the executions in the resources already deployed by Yorc. Based on this scheduling, the COMPSs runtime can interact with the different HPC, DA, and ML runtimes to coordinate resource usage to avoid overlaps and get the maximum performance. Apart from the dynamic task graph generation, the COMPSs runtime is also able to react to task-failures and exceptions in order to adapt the workflow behaviour accordingly. These functionalities, together with similar features provided by Yorc at a higher level, offer the possibility of supporting workflows with a very dynamic behaviour, that can change their configuration at execution time upon the occurrence of given events, such as failures or exceptions [62].

Finally, regarding the integration of the data management and computation, the eFlows4HPC stack provides two solutions for persistent storage: Hecuba (based on key-value databases) and dataClay (object-oriented distributed storage) [63]. These solutions can be used in PyCOMPSs applications to store application objects as persisted objects in new memory devices such as NVRAM or SSDs, enabling the keeping of data after the execution of the application. This changes the paradigm of persistent storage in HPC, dominated by the file system, to other more flexible approaches. By using persisted objects, application patterns such as producer-consumer, in-situ visualisation or analytics, can be easily implemented.

5.3. Architectural Optimizations within eFlows4HPC

One internal and important aspect within eFlows4HPC is the actual performance achieved in the execution of the workflows. Therefore, the project also puts the focus on the identification and optimization of the time-consuming running kernels (understood as independent pieces of code with a well-defined functionality). The optimization process takes into account not only raw performance but also performance-per-Watt. Indeed, new energy-efficient heterogeneous architectures currently being deployed in HPC and DA ecosystems will be targeted devices for the workflow applications. The set of hardware solutions inspected in the project ranges from pre-exascale systems, such as MareNostrum 5, to high-end FPGA devices.

As an example, in the field of workflows using AI-specific components, the project focuses on developing specific kernel optimizations for heterogeneous architectures. One example is the optimization of convolution operations for CNN training and inference. In this direction, a dataflow-oriented programming environment, such as that provided in HLS by Xilinx, enables the design of a pipeline-oriented kernel where throughput is maximized, latency is minimized and, in principle, higher energy efficiency is achieved.

A similar approach is followed in the direction of simulation-oriented HPC applications, specific kernels can be identified and optimized for new emerging technologies, such as RISC-V processors. In that aspect, the European Processor Initiative (EPI) [64] is taken into account with the deployment of RISC-V architectures, emulated on the Marenostrum Exascale Emulation Platform (MEEP) [65].

Finally, GPU-based architectures are also considered used for performance improvement of specific AI-related optimizations, mostly for distributed training as required by the project workflows.

6. Related Work

Section 2 has already introduced part of the related work regarding the topics of the paper. It has introduced the current programming and data management approaches for HPC, DA and ML. The different tools to automate installations and deployments in HPC and Clouds and also some usage models which simplify reusability have been outlined in Section 2.2. One of the models presented in that section is the Function-as-a-Service(FaaS) which offers the functionality implemented in a function without dealing with the deployment and execution details in the Cloud. Our HPCWaaS proposal tries to bring the advantages of the FaaS model into the execution of the complex workflows and HPC environment. However, current commercial or open-source FaaS approaches, which enable deployments on private data centres, are focused on fine-grain computations (duration is restricted to few minutes) and do not support using multiple nodes to host a function execution. Moreover, the HPC environments are restricted environments and system administrators are reluctant to install these kinds of services. For all of these constraints, we propose to build the HPCWaaS offering on top of the proposed software stack designed to operate with HPC environments instead of trying to adapt one of the existing cloud FaaS tools.

Regarding research projects aiming at similar challenges, we can find some EU funded projects dealing with the convergence between HPC and Big Data. For instance, the LEXIS project [66] also tackles the description and automation of computational workflows (e.g., simulations with subsequent data analysis steps) providing easy access to federated computing and data systems. For this purpose, the LEXIS project is building an advanced engineering platform and portal leveraging large-scale geographically distributed HPC and Cloud computing resources. Our approach leverages the results of the LEXIS project, in particular the TOSCA orchestration stack (Yorc). However, we propose a two-level orchestration approach combining TOSCA with a task-based programming model (PyCOMPSs), where TOSCA is used for high-level orchestration and PyCOMPSs for lower-level dynamic workflow execution integrating in-situ AI/ML steps. Moreover, it is bringing more advanced Workflow as a Service features to enable workflow reusability. Similarly, the Evolve project [67] also proposes a software stack to integrate HPC and Big Data environments. In this case, it consists of a Python module for Apache Zeppellin notebook, which is used to create workflows based on container images deployed and orchestrated on top of a Kubernetes architecture. It allows to transparently deploy applications both in the Cloud and in clusters. However, the workflow definition of this approach does not provide the required dynamic behaviour and it relies on Kubernetes, which is rarely found in HPC sites.

The ACROSS project [68] is a European funded project that also aims at combining traditional large scale HPC simulations, high-performance data analytics (HPDA) and machine

learning/deep learning (ML/DL) techniques to boost the performance of the simulation frameworks and/or improve the quality of the simulation results without increasing computing resources consumption. Modern heterogeneous hardware resources ranging from more traditional GPUs, to less common FPGAs and neural network processors (NNPs), could be exploited to efficiently execute such hybrid applications. The goal of the project is to develop an exascale ready, HPC and data-driven execution platform for executing complex workflows of such hybrid applications by focusing on i) the integration of different hardware accelerators, and ii) the implementation of a multi-level, multi-domain orchestrator providing all the mechanisms to easily decompose an application into smaller units of computation, which can be automatically and dynamically mapped on the hardware resources that better fulfil the requirements in terms of performance and energy efficiency.

Regarding the data management challenge, the ADMIRE project (Adaptive multi-tier intelligent data manager for Exascale) [69] focuses on satisfying the performance requirements of today's data processing applications by addressing the storage bottlenecks in HPC architectures. To this aim, the project proposes to create an active I/O stack that dynamically adjusts storage resources to the computational resources of jobs, taking advantage of emerging multi-tier storage hierarchies including persistent memory devices. With an emphasis on storage, the goal of ADMIRE is to improve the performance of HPC applications, and the solution will be evaluated in different domains, including weather, life sciences, physics, remote sensing and deep learning. eFlows4HPC can take advantage of the improvements in the I/O stack resulting from ADMIRE, especially by leveraging the new features and optimizations in the dataClay distributed object store to increase the performance of computational workflows.

Finally, regarding general workflows research activities, it is worth mentioning the WorkflowsRI project [70]. It is an NSF planning grant aiming to engage with representatives from the workflows community including researchers, developers, science and engineering users, and cyberinfrastructure experts. Through targeted community surveys and focused summits, the project will gather a diverse set of perspectives, create a community-owned Workflow Management Systems (WMS) inventory and common knowledge taxonomy, define an experimental methodology for measuring WMS capabilities, and develop a blueprint for a community research infrastructure. The end goal of this project is to design an infrastructure that will have the potential to truly democratize workflows research, enabling researchers, postdocs, and students, irrespective of their institutions, to access cutting-edge infrastructure for comparison, evaluation, and verification of workflows research results.

7. Conclusions

In recent years HPC, as well as DA and AI have evolved providing the user community with powerful tools to develop their applications. However, the lack of programming environments for the development of workflows that include all three aspects is limiting their convergence.

We have identified four main challenges from the areas of application that need to be overcome to achieve this convergence. First, the need for tools that enable the openness, transparency, reusability, and reproducibility of the workflows and their results. Such tools are available in cloud environments but cannot directly be used in HPC systems. Therefore, new tools should be built or adapted from existing ones to provide these functionalities to HPC users, and at the same time complying with the constraints of HPC policies. Second, the development of complex workflows should be made easier while keeping their capabilities and performance. New methodologies that support the development of these workflows are required and simultaneously bridging the gap with the current and future heterogeneous infrastructures. Third, workflow managers must support more dynamicity beyond static pipelines and simple static graphs. The engines should support dynamic shifts in the requested computation according to changes in the input data, computation with urgency demands, and dynamic changes in the workflow executions due to eager analysis of results, exceptions or software faults. What is more, the engine should be able to leverage elastic resource management to deal with changes in the instant workload of the workflow. The fourth challenge comes from the data aspects and their integration with the computation. Similarly to the programming model, HPC and data analytics has been based on different solutions to store the data. Solutions that integrate the different data practices and offer an abstraction layer are necessary. Indeed, with the appearance of new storage devices, the solutions should leverage them and aggregate them in this single data layer.

Taking into account these challenges, we have proposed an architecture for a workflows software stack that offers tools to simplify the development, deployment and execution of the type of complex workflows that we have been describing. In addition to the software stack, the HPC Workflows as a Service (HPCWaaS) paradigm has been proposed as a mechanism to enable reuse, easy deployment, execution and reproduction of the workflow execution. The paradigm has been thought as a mechanism to lower the barriers to the adoption of HPC systems and widen the access to a larger community of users. These ideas are under development in the EuroHPC eFlows4HPC project.

Acknowledgements

This work has been partially supported by the Spanish Government (PID2019-107255GB), by Generalitat de Catalunya (contract 2014-SGR-1051), by German Federal Ministry of Education and Research (16HPC016K), and by the European Commission through the Horizon 2020 Research and Innovation program and the EuroHPC JU under contract 955558 (eFlows4HPC project). The authors also acknowledge financial support from the Spanish Ministry of Economy and Competitiveness , through the "Severo Ochoa Programme for Centres of Excellence in R&D" (CEX2018-000797-S).

References

[1] M. Asch, T. Moore, R. Badia, M. Beck, P. Beckman, T. Bidot, F. Bodin, F. Cappello, A. Choudhary, B. de Supinski, et al., Big data and extreme-scale computing: Pathways to convergence-toward a shaping strategy for a future software and data ecosystem for scientific inquiry, The International Journal of High Performance Computing Applications 32 (4) (2018) 435–479.

[2] Big Data and Extreme-scale Computing web site [cited August, 2021]. URL https://www.exascale.org/bdec/

[3] J. Dongarra, D. A. Reed, Exascale computing and big data, Communications of the ACM 58 (7) (2015) 56–68.

[4] D. Talia, Workflow systems for science: Concepts and tools, ISRN Software Engineering 2013 (2013) 1–15.

[5] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, S. Mock, Kepler: an extensible system for design and execution of scientific workflows, in: Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on, IEEE, 2004, pp. 423–424.

[6] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, et al., The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud, Nucleic acids research 41 (W1) (2013) W557–W561.

[7] E. Afgan, D. Baker, M. van den Beek, D. Blankenberg, D. Bouvier, M. Čech, J. Chilton, D. Clements, N. Coraor, C. Eberhard, B. Grüning, A. Guerler, J. Hillman-Jackson, G. Von Kuster, E. Rasche, N. Soranzo, N. Turaga, J. Taylor, A. Nekrutenko, J. Goecks, The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update, Nucleic Acids Res. 44 (W1) (2016) W3–W10.

[8] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, et al., Pegasus: A framework for mapping complex scientific workflows onto distributed systems, Scientific Programming 13 (3) (2005) 219–237.

[9] T. Fahringer, R. Prodan, R. Duan, F. Nerieri, S. Podlipnig, J. Qin, M. Siddiqui, H.-L. Truong, A. Villazon, M. Wieczorek, Askalon: A grid application development and computing environment, in: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing, IEEE Computer Society, 2005, pp. 122–131.

[10] D. Manubens-Gila, J. Vegas-Regidora, M. C. Acostaa, C. Prodhommea, O. Mula-Vallsa, K. Serradell-Marondaa, F. J. Doblas-Reyes, Autosubmit: a versatile tool for managing Earth system models on HPC platforms, Future Generation Computer Systems submited (2016).

[11] F. Lordan, R. M. Badia, et al., ServiceSs: an interoperable programming framework for the Cloud, Journal of Grid Computing 12 (1) (2014) 67–91.

[12] M. Wilde, M. Hategan, J. M. Wozniak, B. Clifford, D. S. Katz, I. Foster, Swift: A language for distributed parallel scripting, Parallel Computing 37 (9) (2011) 633–652. doi:10.1016/j.parco.2011.05.005.

[13] J. Goecks, A. Nekrutenko, J. Taylor, Galaxy: A comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences, Genome biology 11 (8) (2010) 1–13.

[14] H. J. Oliver, M. Shin, O. Sanders, Cylc: A workflow engine for cycling systems, Journal of Open Source Software 3 (27) (2018) 737.

[15] W. Gropp, W. D. Gropp, E. Lusk, A. Skjellum, A. Lusk, Using MPI: portable parallel programming with the message-passing interface, Vol. 1, MIT Press, 1999.

[16] L. Dagum, R. Menon, Openmp: an industry standard api for shared-memory programming, IEEE computational science and engineering 5 (1) (1998).

[17] Nvidia, CUDA: Compute unified device architecture [cited August,2021]. URL https://docs.nvidia.com/cuda/

[18] F. Marozzo, D. Talia, P. Trunfio, A Workflow Management System for Scalable Data Mining on Clouds, IEEE Transactions on Services Computing 11 (3) (2018) 480–492. doi:10.1109/TSC.2016.2589243.

[19] G. Da Costa, T. Fahringer, J.-A. Rico-Gallego, I. Grasso, A. Hristov, H. D. Karatza, A. Lastovetsky, F. Marozzo, D. Petcu, G. L. Stavrinides, D. Talia, P. Trunfio, H. Astsatryan, Exascale machines require new programming paradigms and runtimes, Supercomputing Frontiers and Innovations 2 (2) (2015) 6–27. doi:10.14529/jsfi150201.

[20] S. Jha, G. Fox, Understanding ML-driven HPC: Applications and infrastructure, in: 2019 15th International Conference on eScience (eScience), 2019, pp. 421–427. doi:10.1109/eScience.2019.00054.

[21] M. Levrier, L. Ganne, F. Exertier, A. Scionti, G. Vitali, O. Terzo, J. Martinovic, M. Golasowski, J. Krenek, F. Donnat, Workflow orchestration on tightly federated computing resources: the LEXIS approach (2020) [cited August , 2021]. URL https://indico.egi.eu/event/5000/

[22] A. Parodi, E. Danovaro, J. Hawkes, T. Quintino, M. Lagasio, F. Delogu, M. D'Andrea, A. Parodi, B. M. Sardo, A. Ajmar, P. Mazzoglio, F. Brocheton, L. Ganne, R. J. García-Hernández, S. Hachinger, M. Hayek, O. Terzo, J. Krenek, J. Martinovic, LEXIS weather and climate large-scale pilot, in: L. Barolli, A. Poniszewska-Maranda, T. Enokido (Eds.), Complex, Intelligent and Software Intensive Systems, Springer International Publishing, 2021, pp. 267–277.

[23] T. Gamblin, M. LeGendre, M. R. Collette, G. L. Lee, A. Moody, B. R. De Supinski, S. Futral, The spack package manager: bringing order to hpc software chaos, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2015, pp. 1–12.

[24] K. Hoste, J. Timmerman, A. Georges, S. De Weirdt, Easybuild: Building software with ease, in: 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, IEEE, 2012, pp. 572–582.

[25] A. Kivity, Y. Kamay, D. Laor, U. Lublin, A. Liguori, kvm: the linux virtual machine monitor, in: Proceedings of the Linux symposium, Vol. 1, Dttawa, Dntorio, Canada, 2007, pp. 225–230.

[26] D. Merkel, et al., Docker: lightweight linux containers for consistent development and deployment, Linux journal 2014 (239) (2014) 2.

[27] G. M. Kurtzer, V. Sochat, M. W. Bauer, Singularity: Scientific containers for mobility of compute, PloS one 12 (5) (2017).

[28] Apache OpenWhisk web site [cited August, 2021]. URL https://openwhisk.apache.org/

[29] OpenFaaS web site [cited August, 2021]. URL https://www.openfaas.com/

[30] J. Lüttgau, M. Kuhn, K. Duwe, Y. Alforov, E. Betke, J. Kunkel, T. Ludwig, Survey of storage systems for high-performance computing, Supercomputing Frontiers and Innovations 5 (1) (2018) 31–58. doi:10.14529/jsfi180103.

[31] H. L. Jenter, R. P. Signell, Netcdf: A freely-available software-solution to data-access problems for numerical modelers, in: Proceedings of the American Society of Civil Engineers Conference on Estuarine and Coastal Modeling, 1992.

[32] M. Folk, G. Heber, Q. Koziol, E. Pourmal, D. Robinson, An overview of the hdf5 technology suite and its applications, in: Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases, AD '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 36–47. doi:10.1145/1966895.1966900.

[33] M. Breitenfeld, N. Fortner, J. Henderson, J. Soumagne, M. Chaarawi, J. Lombardi, Q. Koziol, Daos for extreme-scale systems in scientific applications, ArXiv abs/1712.00423 (2017).

[34] Intel Optane Persistent Memory Workload Solutions [cited August, 2021]. URL https://www.intel.com/content/www/us/en/architecture-and-technology/optane-persistent-memory-solutions.html

[35] J. Nider, C. Mustard, A. Zoltan, A. Fedorova, Processing in storage class memory, in: A. Badam, V. Chidambaram (Eds.), 12th USENIX Workshop on Hot Topics in Storage and File Systems, 2020.

[36] The Green500 list [cited August, 2021]. URL https://www.top500.org/green500/

[37] Google Tensor Flow Processing Unit [cited August, 2021]. URL https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

[38] A. Putnam, A. Caulfield, E. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, E. Peterson, A. Smith, J. Thong, P. Y. Xiao, D. Burger, J. Larus, G. P. Gopal, S. Pope, A reconfigurable fabric for accelerating large-scale datacenter services, in: Proceeding of the 41st Annual International Symposium on Computer Architecuture (ISCA), IEEE Press, 2014, pp. 13–24.

[39] D. Elia, S. Fiore, G. Aloisio, Towards hpc and big data analytics convergence: Design and experimental evaluation of a hpda framework for escience at scale, IEEE Access 9 (2021) 73307–73326. doi:10.1109/

ACCESS.2021.3079139.

[40] V. Eyring, S. Bony, G. A. Meehl, C. A. Senior, B. Stevens, R. J. Stouffer, K. E. Taylor, Overview of the coupled model intercomparison project phase 6 (CMIP6) experimental design and organization, Geoscientific Model Development 9 (5) (2016) 1937–1958. doi:10.5194/gmd-9-1937-2016.

[41] M. Horn, K. Walsh, M. Zhao, S. J. Camargo, E. Scoccimarro, H. Murakami, H. Wang, A. Ballinger, A. Kumar, D. A. Shaevitz, J. A. Jonas, K. Oouchi, Tracking scheme dependence of simulated tropical cyclone response to idealized climate simulations, Journal of Climate 27 (24) (2014) 9197 – 9213. doi:10.1175/JCLI-D-14-00200.1.

[42] ARISTOTLE-eENHSP project web site [cited August, 2021].
URL http://aristotle.ingv.it

[43] A. Grezio, A. Babeyko, M. A. Baptista, J. Behrens, A. Costa, G. Davies, E. L. Geist, S. Glimsdal, F. I. González, J. Griffin, C. B. Harbitz, R. J. LeVeque, S. Lorito, F. Løvholt, R. Omira, C. Mueller, R. Paris, T. Parsons, J. Polet, W. Power, J. Selva, M. B. Sørensen, H. K. Thio, Probabilistic Tsunami Hazard Analysis: Multiple Sources and Global Applications, Reviews of Geophysics 55 (4) (2017) 1158–1198. doi:10.1002/2017RG000579.

[44] G. Davies, J. Griffin, F. Løvholt, S. Glimsdal, C. Harbitz, H. K. Thio, S. Lorito, R. Basili, J. Selva, E. Geist, M. A. Baptista, A global probabilistic tsunami hazard assessment from earthquake sources, Geological Society, London, Special Publications 456 (1) (2018) 219–244. doi:10.1144/SP456.5.

[45] J. Selva, S. Lorito, P. Perfetti, R. Tonini, F. Romano, A. Piatanesi, A. Babeyko, M. Volpe, S. Pintore, M. Mele, A. Amato, Probabilistic tsunami forecasting (PTF) for tsunami early warning operations, in: Geophysical Research Abstracts, Vol. 21, 2019, p. EGU2019–17775.

[46] F. Lovholt, S. Lorito, J. Macías, M. Volpe, J. Selva, S. Gibbons, Urgent Tsunami Computing, in: 2019 IEEE/ACM HPC for Urgent Decision Making (UrgentHPC), IEEE, 2019, pp. 45–50. doi:10.1109/UrgentHPC49580.2019.00011.

[47] T. Goubier, N. Rakowsky, S. Harig, Fast tsunami simulations for a real-time emergency response flow, in: 2020 IEEE/ACM HPC for Urgent Decision Making (UrgentHPC), IEEE, 2020, pp. 21–26.

[48] D. Giles, D. Gopinathan, S. Guillas, F. Dias, Faster than real time tsunami warning with associated hazard uncertainties, Frontiers in Earth Science 8 (2021) 560. doi:10.3389/feart.2020.597865.

[49] J. Selva, A. Amato, A. Armigliato, R. Basili, F. Bernardi, B. Brizuela, M. Cerminara, M. de Micheli Vitturi, D. Di Bucci, P. Di Manna, T. Esposti Ongoro, G. Lacanna, S. Lorito, F. Lovholt, D. Mangione, E. Panunzi, A. Piatanesi, A. Ricciardi, M. Ripepe, F. Romano, M. Santini, A. Scalzo, R. Tonini, M. Volpe, F. Zaniboni, Tsunami risk management for crustal earthquakes and non-seismic sources in italy, La Rivista del Nuovo Cimento 44 (2021). doi:10.1007/s40766-021-00016-9.

[50] D. Salmanidou, S. Guillas, A. Georgiopoulou, F. Dias, Statistical emulation of landslide-induced tsunamis at the Rockall Bank, NE Atlantic, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 473 (2200) (2017).

[51] I. E. Mulia, A. R. Gusman, K. Satake, Applying a deep learning algorithm to tsunami inundation database of megathrust earthquakes, Journal of Geophysical Research: Solid Earth 125 (9) (2020).

[52] F. Makinoshima, Y. Oishi, T. Yamazaki, T. Furumura, F. Imamura, Early forecasting of tsunami inundation from tsunami and geodetic observation data with convolutional neural networks, Nature Communications 12 (1) (2021) 1–10.

[53] J. de la Puente, J. E. Rodriguez, M. Monterrubio-Velasco, O. Rojas, A. Folch, Urgent supercomputing of earthquakes, in: Proceedings of the Platform for Advanced Scientific Computing Conference, ACM, 2020, pp. 1–8. doi:10.1145/3394277.3401853.

[54] J. E. Stone, D. Gohara, G. Shi, Opencl: A parallel programming standard for heterogeneous computing systems, Computing in Science Engineering 12 (3) (2010) 66–73. doi:10.1109/MCSE.2010.69.

[55] S. Wienke, P. Springer, C. Terboven, D. an Mey, Openacc—first experiences with real-world applications, in: European Conference on Parallel Processing, Springer, 2012, pp. 859–870.

[56] J. Reinders, B. Ashbaugh, J. Brodman, M. Kinsner, J. Pennycook, X. Tian, Data Parallel C++: Mastering DPC++ for Programming of Heterogeneous Systems using C++ and SYCL, Springer Nature, 2021.

[57] OASIS, Topology and orchestration specification for cloud applications (2013) [cited August, 2021].
URL http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html

[58] E. Tejedor, R. M. Badia, J. Labarta, et al., PyCOMPSs: Parallel computational workflows in Python, The International Journal of High Performance Computing Applications 31 (2017) 66–82. doi:10.1177/1094342015594678.

[59] L. Richardson, S. Ruby, RESTful web services, " O'Reilly Media, Inc.", 2008.

[60] D. W. Erwin, D. F. Snelling, Unicore: A grid computing environment, in: European Conference on Parallel Processing, Springer, 2001, pp. 825–834.

[61] R. M. Badia, et al., COMP superscalar, an interoperable programming framework, SoftwareX 3 (2015) 32–36. doi:10.1016/j.softx.2015.10.004.

[62] J. Ejarque, M. Bertran, J. Á. Cid-Fuentes, J. Conejero, R. M. Badia, Managing failures in task-based parallel workflows in distributed computing environments, in: European Conference on Parallel Processing, Springer, 2020, pp. 411–425.

[63] J. Martí, A. Queralt, D. Gasull, A. Barceló, J. J. Costa, T. Cortes, Dataclay: A distributed data store for effective inter-player data sharing, Journal of Systems and Software 131 (2017) 129–145. doi:10.1016/j.jss.2017.05.080.

[64] European Processor Intiative web site [cited August, 2021].
URL https://www.european-processor-initiative.eu/

[65] A. Fell, D. J. Mazure, T. C. Garcia, B. Perez, X. Teruel, P. Wilson, J. D. Davis, The marenostrum experimental exascale platform (meep), Supercomputing Frontiers and Innovations 8 (1) (2021) 62–81.

[66] LEXIS project web site [cited August, 2021].
URL https://lexis-project.eu

[67] LEXIS project web site [cited August, 2021].
URL https://www.evolve-h2020.eu

[68] ACCROSS project web site [cited August, 2021].
URL https://www.acrossproject.eu

[69] ADMIRE project web site [cited August, 2021].
URL https://www.admire-eurohpc.eu/

[70] Workflows RI project web site [cited August, 2021].
URL https://workflowsri.org/