DARE: A Reflective Platform Designed to Enable Agile Data-Driven Research on the Cloud

DARE: A Reflective Platform Designed to Enable Agile Data-Driven Research on the Cloud

Iraklis Klampanos*, Athanasios Davvetas*, André Gemünd†, Malcolm Atkinson‡, Antonis Koukourikos*, Rosa Filgueira§, Amrey Krause§, Alessandro Spinuso¶, Angelos Charalambidis*, Federica Magnoni‖, Emanuele Casarotti‖, Christian Pagé**, Mike Lindner†† and Vangelis Karkaletsis*
*Institute of Informatics and Telecommunications, NCSR "Demokritos", Agia Paraskevi, Greece
Email: iaklampanos@iit.demokritos.gr
†Fraunhofer Institute for Algorithms and Scientific Computing, Sankt Augustin, Germany
‡School of Informatics, The University of Edinburgh, Edinburgh, UK
§EPCC, The University of Edinburgh, Edinburgh, UK
¶Koninklijk Nederlands Meteorologisch Instituut (KNMI), Utrecht, The Netherlands
‖Istituto Nazionale Geofisica e Vulcanologia (INGV), Rome, Italy
**Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique (CERFACS), Toulouse, France
††Karlsruher Institut für Technologie (KIT), Karlsruhe, Germany

***Abstract*—The DARE platform has been designed to help research developers deliver user-facing applications and solutions over diverse underlying e-infrastructures, data and computational contexts. The platform is Cloud-ready, and relies on the exposure of API, which are suitable for raising the abstraction level and hiding complexity. It implements the cataloguing and execution of fine-grained and Python-based dispel4py workflows as services. Reflection is achieved via a logical knowledge base, comprising multiple internal catalogues, registries and semantics, while it supports persistent and pervasive data provenance. This paper presents design and implementation aspects of the DARE platform, as well as it provides directions for future development.**

***Index Terms*—software platform, cloud, technology, conceptualization, data-driven science, scientific workflows, provenance, workflow optimization**

I. Introduction

Modern data-driven science takes place on increasingly distributed and diverse infrastructures. Scientific knowledge is therefore scattered across cloud-based services, local storage, and source code targeting specific architectures and computational contexts. Concepts reflected on such disparate sources are hardly computer-communicable and computer-actionable across or even within disciplines. This makes traceability, communication of methods, provenance gathering and reusing data and methods across disciplines harder and more time-consuming. At the same time, commercial clouds play an increasingly important role in large-scale scientific experimentation. Examples of commercial clouds being used in large-scale scientific contexts are found on both sides of the Atlantic: in the European Open Science Cloud[1] (EOSC) case as well as in the massive ongoing migration of data and other resources onto Amazon's AWS by NASA[2].

This work has been supported by the EU H2020 research and innovation programme under grant agreement No 777413.

[1]https://ec.europa.eu/research/openscience/index.cfm?pg=open-science-cloud

[2]https://aws.amazon.com/partners/success/nasa-image-library/

It follows that while potential for large scale data-driven experimentation increases, so does complexity. At the same time, making use of vendor-specific features may lead to lock-in. Users directly being affected by these issues are the research developers: domain experts who develop user-facing solutions on behalf of their communities.

The DARE platform[3], developed as the main technical objective of the DARE project[4], deals with these challenges and helps research developers make better and more transparent use of diverse infrastructures via:

1) The facilitation of high-level programmatic methods via the use of the fine-grained workflow specification library dispel4py [1].
2) Mappings of workflows on different execution contexts within and across cloud deployments.
3) Tools for tracking and analyzing data provenance in real-time.
4) Reflection onto the e-infrastructure via a logical knowledge base, describing the running cloud environment and user information, e.g. workflows, software components, data provenance and data.
5) Integrated big data tools, as well as connectors to external data sources.
6) Exposing all relevant functionality via a set of RESTful APIs that (1) effectively hide technical detail and (2) enable research developers to build solutions that exploit multiple underlying e-infrastructures with minimal effort.

The overarching vision behind DARE as well as its main architectural considerations and components can be found in [2]. This paper describes the current technical instantiation in response to this vision, its main software components and their interactions.

[3]https://gitlab.com/project-dare

[4]http://project-dare.eu

II. Platform Design and Integration

An overview of the platform can be seen in Figure 1. The DARE platform is Cloud-ready and is designed primarily for the currently under-development European Open Science Cloud (EOSC) in mind. This does not preclude it from being readily deployable on other research or commercial Cloud platforms.

The basis of the DARE platform is a big-data integrator platform, which was originally developed as part of the Big Data Europe project [3]. Technologies that co-exist and are integrated on the big-data integrator platform allow for the delivery of the other core elements of the DARE platform, such as the knowledge base components, the optimizing workflow system and methods, the provenance subsystem, and so on.

A. Cloud-Ready Platform

The DARE platform relies on the integration of containerized software components. Containerized applications enable software isolation with no impact on the application performance. Containers share resources with the host operating system and enable consistent development [4]. Containerized applications are supported by cloud infrastructure providers and are considered as cloud-native.

The Kubernetes[5] orchestration enables the effortless cloud deployment of the DARE platform. Kubernetes enables automated deployment, scaling and management of containerized applications. Cluster management and deployment is operated through its exposed API. Kubernetes API enables external and internal communication exploiting user-client authentication. Kubernetes API additionally provides Role-Based Access Control (RBAC) [5] that binds a user-client (which can be a containerized application) to the cluster. Using RBAC makes user authentication and access flexible and manageable by enabling permission control for Kubernetes resources. Furthermore, Kubernetes can be enriched by native add-ons such as DNS or GUI dashboards, as well as, Custom Resource Definition of Services and other Kubernetes resources that provide additional functionality.

[5]https://kubernetes.io/

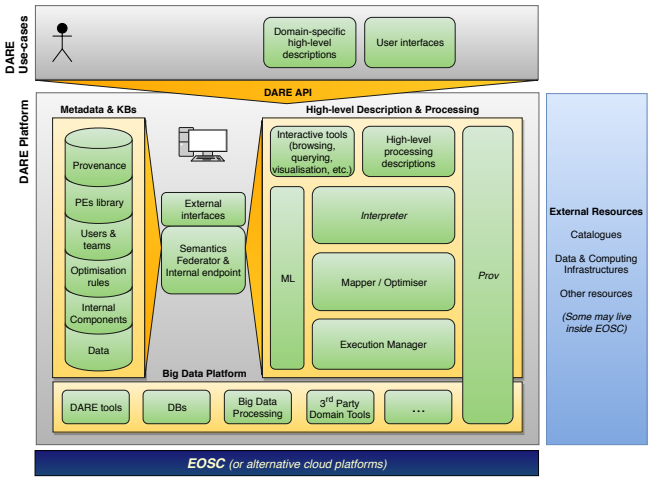

Fig. 1. Overview of the DARE platform.

Through its deployment, the DARE platform can make use of Big Data Processing tools provided by the BigDataEurope platform [6]. The Big Data Europe platform includes cloud-native and ready to deploy containerized versions of big data tools such as Apache Hadoop, Apache Kafka, the PostgreSQL database, Flink, etc. Intuitively, any cloud-native containerized software component such as Databases or Third party domain tool that is necessary for DARE use-cases can be deployed as a part of the DARE platform and can be accessed by its internal components.

1) Provisions for the EOSC: The DARE platform has been designed to accommodate the European Open Science Cloud on two levels. On one hand, the platform and services have been designed to complement existing services offered through EOSC instead of duplicating work. For example, for long-term storage, it is designed to interface to the EUDAT B2SAFE service[7]. On the other hand, the stack has been designed and packaged so that it is easily deployable on EOSC infrastructure services. To achieve the latter goal, DARE offers ready-to-use infrastructure descriptions and deployment recipes based on the well-known Ansible by Red Hat[8] and Terraform by HashiCorp[9]. Terraform is used to automatically deploy a set of Virtual Machines with the required infrastructure properties such as CPU, memory, storage and network interfaces on an IaaS Cloud of choice. Supported Cloud backends include e.g. AWS, GCP, Microsoft Azure and OpenStack[10].

DARE primarily targets the EGI Federated Cloud[11], which is part of the EOSC and offers access to the OpenStack API on the sites of its federation. After the Virtual Machines have been started, Ansible installs and configures the Kubernetes stack, and starts complementing add-ons, such as networking (based on Calico[12]) and storage (based on Rook[13] Ceph). The DARE stack running on top of Kubernetes in turn makes extensive use of Helm Charts to package and manage the Kubernetes resources and applications. This also facilitates usage on existing Kubernetes clusters.

Furthermore, to seamlessly integrate with EOSC, the platform's authentication and authorization mechanisms have been designed to be interoperable with existing Authentication and Authorization Infrastructure (AAI). To that end, the DARE platform employs OAuth2 with OpenID Connect. The same technology is the basis for the EOSC portal[14], EGI Check-In[15] and EUDAT B2Access[16]. Through the use of an own Key-

[6]https://github.com/big-data-europe
[7]https://www.eudat.eu/b2safe
[8]https://www.ansible.com/
[9]https://www.terraform.io/
[10]https://www.openstack.org
[11]https://www.egi.eu/federation/egi-federated-cloud/
[12]https://projectcalico.org
[13]https://rook.io/
[14]https://eosc-portal.eu/
[15]https://access.egi.eu/
[16]https://b2access.eudat.eu/

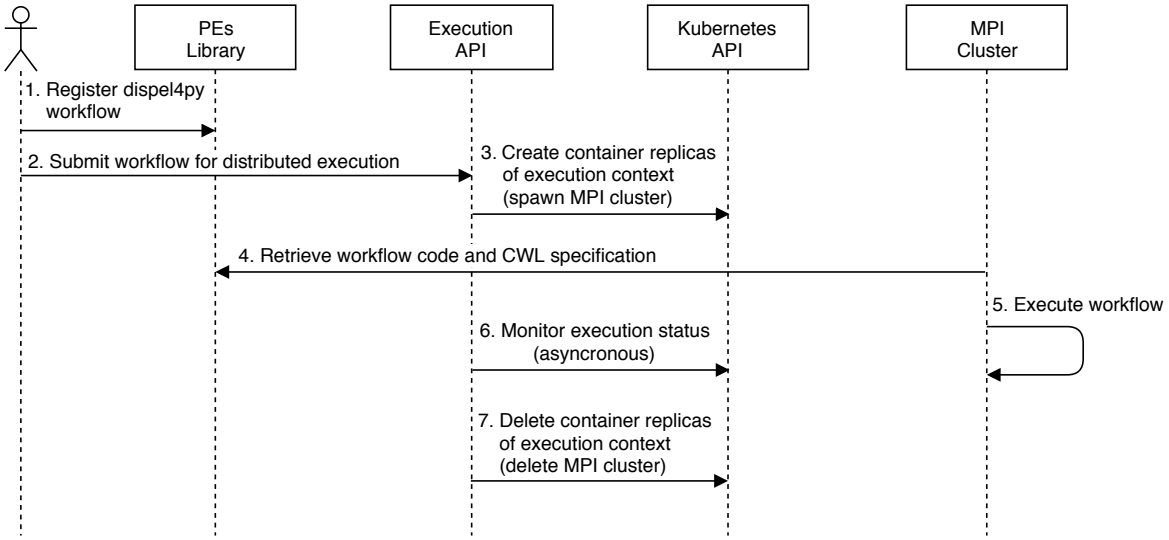

Fig. 2. The executions of a dispel4py workflow using the DARE API. This figure shows all the underlying steps as well as the entities that are involved during execution. Steps 1, 2 and 4 are HTTP calls. Steps 3, 6 and 7 are executed within python, while step 5 is executed using CWL.

cloak[17] deployment, the DARE platform allows community administrators the choice to implement own identity databases or integrate with all providers or proxies that implement the OAuth2 with OpenID Connect technology. These include the ones mentioned above as well as many other providers, such as Google, Facebook, Github, etc.

The DARE implementation makes use of Kubernetes sidecar containers based on the Keycloak Gateway, which are automatically injected with application containers on Kubernetes pods and act as a reverse proxy to the application. Connections to the service go to the sidecar proxy first, which checks if the user is already authenticated and if not sends a forward header to the login page. If the user is authenticated, the request is forwarded to the service with additional HTML headers injected, which allow the application to identify the user. The application only has to worry about the identity, while the proxy deals with session identifiers, invalid sessions and tokens. The project is currently setting up a demonstration platform which will make use of EOSC Cloud resources and will use an integrated AAI.

B. Workflows-as-a-Service and the DARE API

DARE API is a composition of RESTful Web APIs exposed by the containerized versions of the underlying DARE platform components. Exposing the Web APIs of otherwise isolated software components of DARE platform enables across component communication, as well as, its coupling with any user interface appropriate for DARE use-cases. Essentially, the DARE API acts as a gateway between the cloud deployed DARE platform components and the interactive user interfaces.

During its first iteration DARE API consists of the *Execution API* and *PEs library API* to allow for a *Workflows-as-a-Service* functionality in DARE's pilot use-cases, as well as the *s-provflow* that provides Provenance. Execution API is mainly responsible for the distributed and scalable execution of Dispel4Py workflows [1] or Specfem3D simulations [6]. The Execution API allows the submission of execution jobs through exposed web service resources. Therefore, allowing for scalable integration of additional execution contexts, since the individual execution contexts are containerized components instantiated from the Execution API.

For the purposes of scalable and distributed execution, Execution API is deployed with "unlimited access" role to the Kubernetes resources. During the submission of any job the user can specify the amount of nodes or workers that are required for the distributed execution of their workflow. Execution API spawns replicas of the specified execution context in order to be used within a Master/Worker model. Additionally, it launches an asynchronous function that monitors the status of running jobs and redirects their logs. After the end of the execution the container cluster of replicas is deleted to release the occupied resources. The above steps can be also observed in Figure 2 and Figure 3.

For all execution contexts the Common Workflow Language (CWL) [7] is utilized during run time. CWL is a workflow specification that besides workflow description allows for transparency and standardization. In the execution contexts of DARE API, CWL allows the dynamic parameterization of executions. In the Specfem3D context, CWL is used as a higher level workflow description. It describes the steps that are necessary in order to perform a single simulation, the input and output of each step, the running order of steps and the scripts that have to be executed. On the other hand, on the context of executing dispel4py workflows, CWL is used to describe and dynamically parameterize distributed dispel4py execution on command level.

Post execution logs and outputs are stored within a shared

[17]https://www.keycloak.org

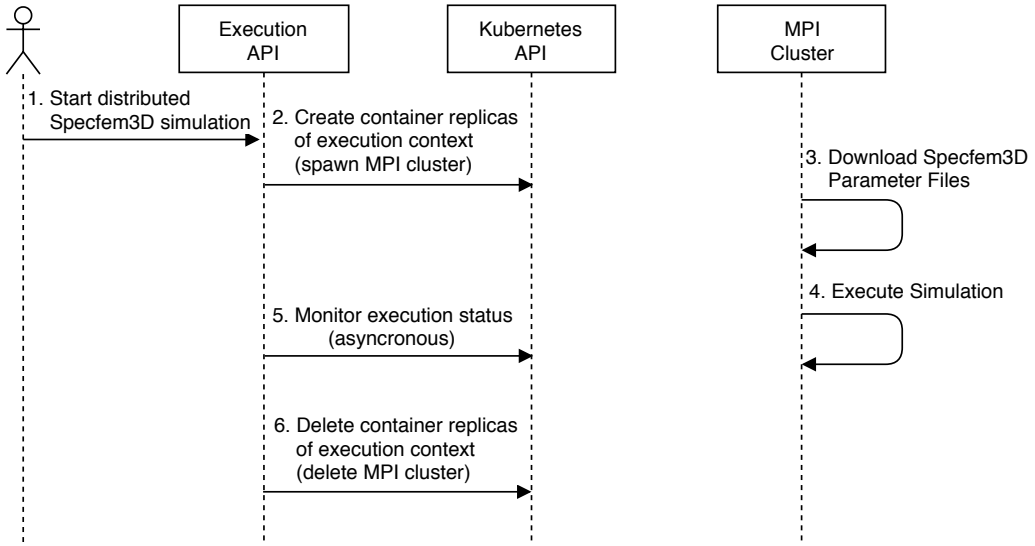

Fig. 3. Specfem3D waveform simulation execution overview using DARE API. This figure depicts all the underlying steps as well as the entities that are involved during execution. Step 1 is an HTTP call. Steps 2, 5 and 6 are executed within python and steps 3 and 4 are part of the CWL workflow execution.

file system between the Execution API and all spawned container clusters using the cloud-native storage Rook-Ceph[18]. The Execution API creates a folder structure for each run in order to store generated outputs. Users of DARE platform can interact with the shared file system through the Execution API in order to upload input files, to review generated outputs or download files. Users also have the choice to download and reference files through the Execution API by using the B2DROP service of EUDAT[19].

PEs library provides the functionality of registering workflow entities, such as processing elements (PEs), functions and literals. It aims at storing and presenting information regarding workflows. A processing element is a single computational unit that is utilized inside dispel4py in order to create workflow graphs. The PEs library facilitates the storing of code implementation for such computational units to enable the sharing and collaboration. For that purpose, it provides workspace structures for registering workflow entities that enable user individuality. It additionally, provides transparency and reproducibility of workflow executions.

Furthermore, PEs library is a manageable way to allow users to execute arbitrary code within the DARE platform. Arbitrary workflow execution raises multiple security concerns. In cases of suspected malicious activity, the PEs library can be utilized to monitor the workflow registration and execution activity. Despite that, the workflows are executed in temporary container clusters with no public entry points or cloud access after their registration.

C. Knowledge Base, Provenance Tracking and Metadata

As per its reference architecture [2], the DARE platform implements a logical knowledge-base as a series of stores and registries. The semantification and tighter integration of these constituent stores and registries is ongoing work.

The current implementation of the DARE knowledge-base comprises the following components:

1) Workflows and PEs Registry: This component is designed to primarily describe dispel4py workflows and processing elements. It exposes a RESTful API and is able to manage workspaces in order to isolate work between users, while at the same time to allow collaboration and exchange of methods and ideas. DARE requires a workflow to be registered before it can be executed. This enforces a checkpoint and helps to avoid executing arbitrary and unchecked code on the platform. The current implementation is based on a relational schema. An introduction to its data model, entities and intended use can be found in [8].
2) The Provenance store: This component is the store for all provenance information gathered within DARE. Provenance is a big part of DARE platform, with the component S-ProvFlow being natively supported by dispel4py. S-ProvFlow continuously captures data provenance associated with: computation unit distribution and parallelization, run-time changes, metadata and generated data products. It enables the reproduction of generated data outputs and real-time monitoring of resource use. The current implementation is based on MongoDB in order to allow responsiveness while also being flexible with respect to the data model.
3) Internal components catalogue: This catalogue captures information regarding the available components that are deployed in the cloud environment. The metadata captured in the internal components catalogue are exploited for informed resource acquisition as well as optimized execution context selection. The current implementation

[18]https://rook.io/
[19]https://eudat.eu/services/b2drop

is currently based on the Kubernetes API, with future work directed on improving and extending it.

4) Data catalogue: The main purpose of the data catalogue is to provide a list of internal to DARE, as well as external datasets along with their semantics (e.g. domain-specific descriptions and interlinking). Additionally, it should provide an additional level of abstraction between conceptual descriptions and their digital representations (e.g. formats, data types, locations, etc.). The data catalogue should enable collaboration and sharing of reusable data products. The current implementation is based on linked data technologies, however it covers a small fraction of the target requirements with future work being directed at improving and extending it.

In addition to the registries and catalogues described above, DARE may require additional components in order to cover emerging requirements of optimizing workflows, identifying users, anonymizing data and metadata, etc. (Figure 1).

III. An End-to-End Example Use-Case

In this section we present a simple but complete use-case of using the DARE platform in order to execute a workflow in a scalable, flexible and distributed manner.

The executed workflow is described in Algorithm 1. In this simple use case, we create a computational graph consisting of four processing elements. *Split* processing element distributes a list of number into equal parts, for uneven divisions between length of list and workers is being compensated by replacing it with the smallest integer value. *Mult* is a function that multiplies the elements of a list by two. Each worker independently calls this function on its part of the split list. The *Merge* processing element merges back all the multiplied parts from the individual workers into a new single list of elements. The *Fwrite* processing element simply writes the contents of a list into a text file.

In the case the PEs above are not already available in the DARE Workflows and PEs registry, to execute this simple use-case using the DARE API one must follow the steps below. (Information on the Workflows and PEs registry entities and intended use can be found in [8].)

1) Authenticate against the Workflows and PEs registry (PEs library API)
2) Create a workspace on PEs library (PEs library API)
3) Create processing element signature of the workflow (PEs library API)
4) Create processing element implementation of the workflow - Register workflow implementation source code (PEs library API)
5) Submit registered workflow for execution (Execution API)

Additionally, a user might want to run these optional steps:

1) Monitor status of spawned container cluster (Execution API)
2) List user directories (Execution API)
3) List files within certain directory (Execution API)
4) Download / Get B2DROP link of generated output text file (Execution API)

Algorithm 1 Simple use case

Input: Python list of numbers
Output: List of numbers multiplied by 2, text file

```
function splitPE(list):
    Get input list, split list
    into equal length parts
function mult(list):
    return list numbers multiplied by 2
function mergePE(parts of list):
    merge list parts into a single list
function fwritePE(list):
    write elements of list in file
Initialize list: numbers
Graph:
    splits = splitPE(numbers)
    mult(splits)
    new_numbers = mergePE(splits)
    fwritePE(new_numbers)
Execute Graph using dispel4py
```

In Figure 4 we illustrate the interactive visualization of the lineage recorded during the execution of the *Split and Merge* workflow and offered by the S-ProvFlow component. This Figure also illustrates the dynamic creation of PE instances, in this case making use of MPI parallelization and the dispel4py corresponding mapping.

IV. Scientific Use-Cases

In this section we provide the description of two scientific pilot use-cases the DARE platform directly responds to. These use-cases are being developed as part of the DARE project and are described in more detail in [2].

A. Seismology

The first use-case is on seismology, where we consider the case of ground motion Rapid Assessment (RA). The objective of this use-case is to analyze seismic wavefields immediately after the occurrence of large earthquakes and produce in-time, on-demand estimates of ground motion parameters as peak values of ground velocity or acceleration. The Rapid Assessment use-case can be useful during emergency contexts in order to create maps that will compare observation data along with synthetic data in order to get a better understanding of the ground behaviour.

For the Rapid Assessment use-case, the first step is to run the Specfem3D waveform simulation based on a provisional 1D Time Domain Moment Tensor (TDMT) source solution in order to generate synthetic data. The following steps after waveform simulation are:

1) download observed data for the chosen earthquake from stations through the Federation of Digital Seismographic Network (FDSN)

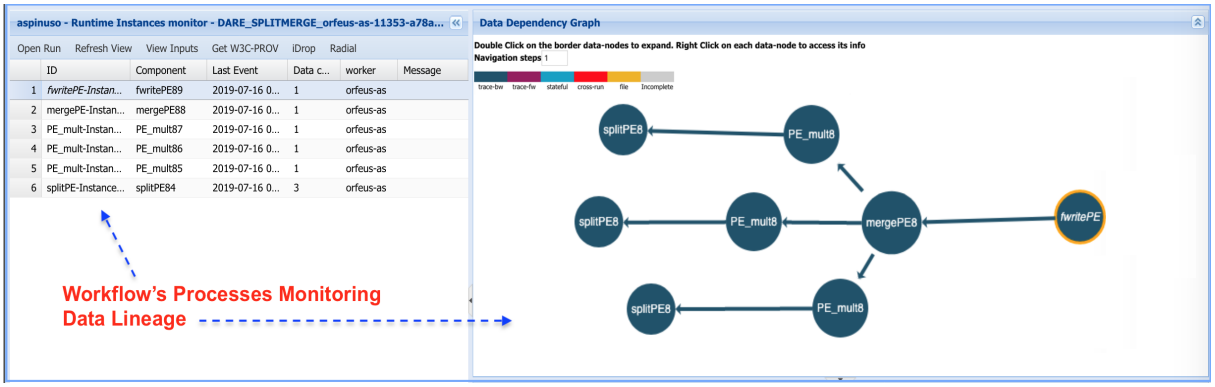

Fig. 4. Provenance visualization of the execution of the *Split and Merge* workflow in *S-ProvFlow*. The left panel shows a runtime monitor of the workflow's processes. For each instance of a PE, the interface displays the timestamp of the last data produced, the worker node where the PE runs, the count of the generated data and the occurrence of system messages. The right panel instead offers the possibility to analyze the data-lineage associated with a particular PE output (circles). Arrows represent the provenance relationship *wasDerivedFrom*. Yellow circles indicate that the data is materialized and can be accessed at a specific location. For visual compression the circles' labels show the name of the PE generating the data. Users interacts with the graph in order to access detailed metadata about the data and its generating process.

2) pre-process synthetic and observed seismograms
3) calculate and compare ground motion parameters between observed and synthetic data, and
4) plot ground motion parameters for both data

In DARE, all the aforementioned steps are implemented as dispel4py workflows, which means that they have to be preceded by API calls used to authenticate and create the appropriate entries in the Workflows and PEs registry. Then, each step-workflow can be executed via the Execution API.

B. Climate Change Impact

Climate4Impact[20] is a platform of services providing also a web front-end. It is aimed at climate change impact modellers, impact and adaptation consultants, other researchers using climate data. It has been developed in the EU funded IS-ENES projects since 2009. The objective of Climate4Impact is to enhance and open up the use of climate modelling data for research. Further, it allows for visualization and download of data from global climate models, regional climate models and downscaled high resolution climate data. It also provides on-demand downscaling, subsetting, regridding and processing of data. The purpose of this use-case is to enhance, accelerate and ensure scalability of users' on-demand calculations using the Climate4Impact platform, by delegating those to the DARE platform and bringing back the results to the Climate4Impact front-end. This delegation has to be transparent to the end users.

The following is done transparently by the Climate4Impact platform itself, taking input from the front-end parameters and workflows specified by the end users. It first needs to authenticate and register the necessary workflows in the Workflows and PEs registry. The Climate4Impact processing service then uploads a json file in the shared file system which is referenced within the workflow as its input. The workflow aims at providing access to and reducing NetCDF files obtained from Earth System Grid Federation (ESGF). It also performs post-processing analysis to prepare data for potential further processing or presentation to end-users. Once the workflow has been registered with the DARE platform it is executed via the Execution API, on behalf of the user.

[20]https://climate4impact.eu

V. Related Work

The DARE platform further extends tools and advances made in earlier projects. It integrates technologies in ways that enable new types of interaction with the underlying systems and e-infrastructures, making it easier for research developers and scientists to describe and execute data-driven computational experiments. Workflow-as-a-Service components dispel4py [1], s-ProvFlow [9] and the dispel4py PEs and workflows library[21] were originally developed as part of the VERCE project[22] [10]. One of VERCE's final deliverables was a workflow-driven e-Science gateway for seismology. The VERCE gateway was targeted at the Seismology scientist and offered largely fixed functionality. DARE exploits these technologies, integrating them further and collectively exposing them via its APIs to offer workflows-as-a-service functionality to research developers.

The BigDataEurope integrator platform [3] was designed in order to democratize big data technologies and to expose them to policy makers and to the general public. The integration and testing of the platform was driven and evaluated via a series of use-case pilots that covered all seven societal challenges set out at the time by the European Commission[23]. The DARE platform assimilates know-how and technology from the BigDataEurope platform as its technological basis on top of which the complete solution is integrated.

[21]VERCE Architecture and Tools for Data-Intensive Applications: http://www.verce.eu/Repository/Deliverables/RP3/D-JRA2.1.2.pdf

[22]http://verce.eu

[23]https://ec.europa.eu/programmes/horizon2020/en/h2020-section/societal-challenges

The DARE project and platform aims to be readily deployable on Cloud platforms, with a specific focus on the European Open Science Cloud (EOSC)[24]. EOSC is a loose federation of cloud resources, providers and policies that offers domain-specific and generic services to scientists and the general public. As integration with EOSC is of priority, the DARE platform has been designed with input from influential initiatives that shape the presence and future of EOSC, such as EOSC-Hub[25], OpenAIRE[26], and others.

VI. Conclusions and Future Work

In this paper we presented the design, implementation and integration of core technologies in the DARE platform. DARE has been designed to help research developers deliver high-quality user-facing applications to their target scientific domains. DARE is Cloud-ready and exploits technologies such as containerization and cataloguing in order to execute high-level, fine-grained workflows on behalf of its users. The use of such workflows, as well as other functionality that is exposed via Web APIs allow for the hiding of details of underlying e-infrastructures, data and computational contexts, as well as it protects research developer from getting locked into vendor-specific technologies. The presence of persistent and pervasive provenance allows for research results to be communicable and reproducible.

In the near future we aim to improve the DARE platform in a number of ways, including: (1) providing smart enactment over DARE-managed resources on the Cloud, (2) extending and unifying the internal catalogues and registries in order to reach a more consistent view of the DARE knowledge base, (3) improving internal data representation and resolution and (4) further integration with 3rd-party e-infrastructures and platforms.

References

[1] R. Filguiera, A. Krause, M. Atkinson, I. Klampanos, and A. Moreno, "Dispel4py: A Python framework for data-intensive scientific computing," International Journal of High Performance Computing Applications, 2017.

[2] M. Atkinson, R. Filgueira, I. Klampanos, A. Koukourikos, A. Krause, F. Magnoni, C. Pag, A. Rietbrock, and A. Spinuso, "Comprehensible control for researchers and developers facing data challenges," in Proceedings of the 15th IEEE International Conference on eScience (to appear), 2019.

[3] S. Auer, S. Scerri, A. Versteden, E. Pauwels, A. Charalambidis, S. Konstantopoulos, J. Lehmann, H. Jabeen, I. Ermilov, G. Sejdiu, A. Ikonomopoulos, S. Andronopoulos, M. Vlachogiannis, C. Pappas, A. Davettas, I. A. Klampanos, E. Grigoropoulos, V. Karkaletsis, V. de Boer, R. Siebes, M. N. Mami, S. Albani, M. Lazzarini, P. Nunes, E. Angiuli, N. Pittaras, G. Giannakopoulos, G. Argyriou, G. Stamoulis, G. Papadakis, M. Koubarakis, P. Karampiperis, A.-C. N. Ngomo, and M.-E. Vidal, "The BigDataEurope Platform - Supporting the Variety Dimension of Big Data," in 17th International Conference on Web Engineering (ICWE2017), 2017. [Online]. Available: http://jens-lehmann.org/files/2017/icwe_bde.pdf

[4] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," Linux J., vol. 2014, no. 239, Mar. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2600239.2600241

[5] D. Ferraiolo and R. Kuhn, "Role-based access control," in In 15th NIST-NCSC National Computer Security Conference, 1992, pp. 554–563.

[6] D. Peter, D. Komatitsch, Y. Luo, R. Martin, N. Le Goff, E. Casarotti, P. Le Loher, F. Magnoni, Q. Liu, C. Blitz, T. Nissen-Meyer, P. Basini, and J. Tromp, "Forward and adjoint simulations of seismic wave propagation on fully unstructured hexahedral meshes," Geophysical Journal International, vol. 186, no. 2, pp. 721–739, 08 2011. [Online]. Available: https://doi.org/10.1111/j.1365-246X.2011.05044.x

[7] P. Amstutz, M. R. Crusoe, N. Tijani, B. Chapman, J. Chilton, M. Heuer, A. Kartashov, D. Leehr, H. Mnager, M. Nedeljkovich, and et al., "Common workflow language, v1.0," Jul 2016. [Online]. Available: https://figshare.com/articles/Common_Workflow_Language_draft_3/3115156/2

[8] M. Atkinson, M. Galea, and I. Klampanos, "D-JRA2.1.2: VERCE Architecture and Tools for Data-Intensive Applications," Tech. Rep., 2013. [Online]. Available: http://verce.eu/Repository/Deliverables/RP3/D-JRA2.1.2.pdf

[9] A. Spinuso, "Active provenance for data-intensive research," Ph.D. dissertation, The University of Edinburgh, 2018.

[10] M. Atkinson, M. Carpene, E. Casarotti, S. Claus, R. Filgueira, A. Frank, M. Galea, T. Garth, A. Gemund, H. Igel, I. Klampanos, A. Krause, L. Krischer, S. H. Leong, F. Magnoni, J. Matser, A. Michelini, A. Rietbrock, H. Schwichtenberg, A. Spinuso, and J. P. Vilotte, "VERCE delivers a productive e-science environment for seismology research," in Proceedings of the 11th IEEE International Conference on eScience, 2015.

[24] https://www.eosc-portal.eu

[25] https://www.eosc-hub.eu

[26] https://www.openaire.eu