DARE Platform: Enabling Easy Data-Intensive Workflow Composition and Deployment

Rosa Filgueira
EPCC
The University of Edinburgh
Edinburgh, UK
rosa.filgueira@ed.ac.uk

***Abstract*—This work presents the DARE platform and working environment for enabling easy data-intensive workflow composition and deployment on clouds systems. DAREs technology translates scientists methods to concrete scientific workflows that can be portable and reproducible on different computing environments without making any (or little) changes. For achieving this, we have combined the strengths of dispel4py and CWL scientific workflows, Docker containers, Kubernetes infrastructure orchestration, Jupyter notebooks, and Cloud platforms.**

***Index Terms*—Scientific Workflows, Cloud systems, FAIR**

I. Introduction

Virtually every domain is enjoying an increasing wealth of data, e.g., from observing natural phenomena, experiments, societal behavior, or business transactions. The complexity of todays challenges and the increases in computational power lead to simulations that yield large volumes of data. Models and understanding are improved by comparing such results with observationsa demanding data-intensive stage in many scientific methods.

There is commensurate growth in expectations about what can be achieved with this wealth of data and computational power. To meet these expectations with available expertise requires tools that make it far easier to reliably formalise data-driven methods that exploit high-end architectures efficiently to meet the needs of science. Therefore, DARE[1] project focuses on empowering domain experts to invent and improve their methods and models by providing a new platform and a working environment. DAREs technology will translate the scientists methods to concrete applications that will be deployed and executed on cloud resources.

DARE has initially focused in two scientific communities: Seismology (EPOS [2] and Climate (IS-ENES[3] Research Infrastructure). In this work we describe the advance interfaces developed to support the Rapid Ground Motion Assessment (*RA*) application (see Figure 1). It requires rapid data analyses, handling multiple data formats, multiple data sources, and availability of computing and storage resources on demand.

Our main objective here was to build this application in such a way that can be portable and reproducible [1] without making any (or little) changes if we run it on different computing resources. For doing so, we have exploited different technologies, CWL [2] and dispel4py [3] workflows, Docker containers, Kubernetes infrastructure orchestrations, Jupyter notebooks, and Cloud platforms.

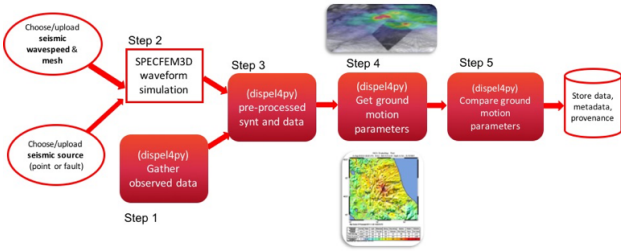

Fig. 1: Rapid Ground Motion Assessment application (*RA*.

RA aims to model the strong ground motion after large earthquakes, in order to make rapid assessment of the earthquakes impact, also in the context of emergency response. It has five main phases: (1) to select an earthquake gathering the real observed seismic wavefield, (2) to simulate synthetic seismic waveforms corresponding to the same earthquake using SPECFEM3D [4], a MPI-based parallel software; (3) to pre-process both synthetic and real data; (4) to calculate the ground motion parameters for synthetic and real data; (5) to compare them with each other by creating shake maps (see Figure 2)

First, we built a dispel4py [5] workflow to represent each part of the RA as a streaming pipeline application (see Figure 3), except for the generation of the synthetic data, since SPECFEM3D is a MPI parallel application on its own. dispel4py is a stream-based data pipeline framework, that allows for developing applications on local machines and run them at distributed computing resources, such Clouds and HPC clusters, without making changes. Users just need to express their computational activities in an abstract way, thinking just how to connect them. One of the key-features of dispel4py is that it provides automatic mappings to different parallel engines, such as MPI or Apache-Storm, as well as a sequential mapping to test applications locally.

Then, we combined the strengths of dispel4py with CWL, which is a specification for describing the data and execution model of workflows/command tools. We used CWL to connect RA dispel4py workflows, to describe semantically their input

[1]http://project-dare.eu/
[2]https://www.epos-ip.org/
[3]https://is.enes.org/

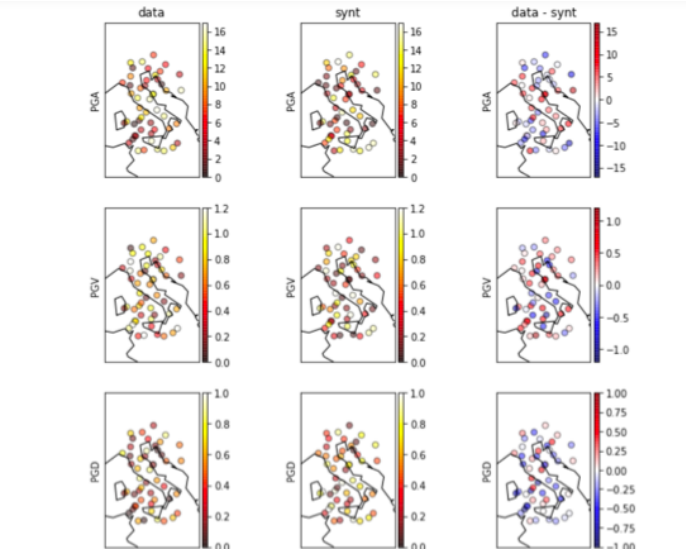

Fig. 2: Shake maps of ground motion parameters are fundamental for a visual representation of the earthquake.

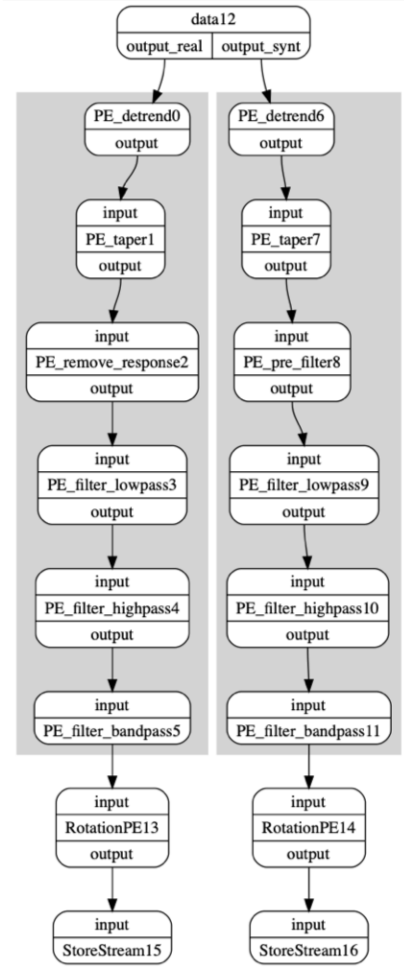

Fig. 3: dispel4py pre-processing workflow, which corresponds to the third step of the RA.

and output parameters, and to orchestrate their executions by using different mappings based on the computing resource available at each time.

Later, we created a new docker container with SPECFEM3D and an MPI cluster[4] , which allows for generating synthetic waveforms using either local or distributed resources. CWL was also used here for describing and managing the execution of SPECFEM3D, enabling us to fully connect all the RA steps.

For validating all of our work we first run all the steps of the *RA* in our laptops, using a small dataset and the sequential dispel4py mapping for all the workflows. Once validated the results, we executed the same codes (dispel4py workflows and SPECFEM3D) using the NSF-Chameleon [5]cloud (using a VM with 24 cores), with a larger dataset, and the parallel MPI dispel4py mapping. In both execution environments, CWL was in charge to execute and connect each part of the RA application.

Results shown that combining CWL, dispel4py and SPECFEM3D container, we are able to manage the entire application without users making changes to run it on different computing resources, as well as scale it up automatically.

Therefore, the new interfaces that we are building on DARE provide a fluent path from prototyping to production. Applications are not locked to platforms but can be moved to suitable new platforms without human intervention and with the encoded methods semantics unchanged.

In the future, DARE platform will act as an intermediary between users applications and the underlying computing resources, making use of the technologies described before. An API is being developed through which *RA* application can be submitted, and it deploys automatically the necessary environment on demand to run and monitor the application, as well collecting its provenance and results.

References

[1] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al., The fair guiding principles for scientific data management and stewardship, Scientific data 3. doi:10.1038/sdata.2016.18.

[2] P. Amstutz, M. R. Crusoe, N. Tijani, B. Chapman, J. Chilton, M. Heuer, A. Kartashov, J. Kern, D. Leehr, H. Mnager, M. Nedeljkovich, M. Scales, S. Soiland-Reyes, L. Stojanovic, Common workflow language, v1.0 (2016). doi:10.6084/m9.figshare.3115156.v2.

[3] R. Filguiera, A. Krause, M. Atkinson, I. Klampanos, A. Moreno, dispel4py: A python framework for data-intensive scientific computing, The International Journal of High Performance Computing Applications 31 (4) (2017) 316–334. doi:10.1177/1094342016649766.

[4] D. Peter, D. Komatitsch, Y. Luo, R. Martin, N. L. Goff, E. Casarotti, P. L. Loher, F. Magnoni, Q. Liu, C. Blitz, T. Nissen-Meyer, P. Basini, J. Tromp, Forward and adjoint simulations of seismic wave propagation on fully unstructured hexahedral meshes, Geophys. J. Int. 186 (2011) 721–789.

[4]https://gitlab.com/project-dare/WP6_EPOS/tree/master/specfem3d/docker

[5]https://www.chameleoncloud.org/