H2020-EINFRA-2017

EINFRA-21-2017 - Platform-driven e-infrastructure innovation

DARE [777413] "Delivering Agile Research Excellence on European e-Infrastructures"

D4.4 Data-driven Abstraction Specification and Execution Mapping Services Toolkit II

Project Reference No	777413 — DARE — H2020-EINFRA-2017 / EINFRA-21-2017
Deliverable	D4.4 Data-driven Abstraction Specification and Execution Mapping Services Toolkit II
Work package	WP4: Big Data Processing and Analytics
Tasks involved	T4.2, T4.4
Type	DEM: Demonstrator, pilot, prototype
Dissemination Level	PU = Public
Due Date	31/12/2020
Submission Date	30/12/2020
Status	Draft
Editor(s)	Malcolm Atkinson (UEDIN)
Contributor(s)	Rosa Filgueira (UEDIN)
Reviewer(s)	Stavros Sachtouris (GRNET)
Document description	The report accompanies the software deliverable for the Abstraction Specification & Execution Mapping components integrated in the DARE platform. It provides a short description on the different components, their role within the DARE platform and their current status of maturity. It also provides links to relevant code and documentation.

Document Revision History

Update	Date	Modifications Introduced	
		Modification Reason	Modified by
1	7/12/2020	Copy of D3.6 as outline	Malcolm Atkinson (UEDIN)
2	16/10/2020	Report on dispel4py optimisation	Rosa Filgueira (UEDIN)
3	16/12/2020	Editorial pass and insert of above	Malcolm Atkinson (UEDIN)
4	28/12/2020	Response to review	Malcolm Atkinson (UEDIN)

Executive Summary

This task is responsible for designing, implementing and iteratively enriching the execution mapping services foreseen in DARE. The set of services will integrate the interfacing with different execution platforms and frameworks, continuously supporting further processing assets, in accordance with the priorities and needs posed by the user communities.

This deliverable outlines the data-driven abstraction specification DARE software components produced by the extension and improvement of dispel4py and inclusion of CWL. These components allow for the context-agnostic, abstract specifications of methods addressing data, computing and complexity extremes.

New features were added to dispel4py to extend the support for distributed mappings and more functionality as processing element implementations. Dynamic optimisation was introduced.

CWL was included within the framework.

The abstractions defined by dispel4py and CWL are exposed through the DARE API. The DARE API is the entry point of DARE that provides access to the computational resources of the platform. A library of Python functions is provided to help developers use the API.

Table of Contents

1 Introduction

1.1 Purpose and Scope

This task is responsible for designing, implementing and iteratively enriching the execution mapping services foreseen in DARE. The set of services will integrate the interfacing with different execution platforms and frameworks, continuously supporting further processing assets, in accordance with the priorities and needs posed by the user communities.

This deliverable outlines the data-driven abstraction specification DARE software components produced by the extension and improvement of dispel4py and the addition of CWL. These components allow for the context-agnostic, abstract specifications of methods addressing data, computing and complexity extremes.

1.2 Approach and relationship with other Work Packages and Deliverables

The architecture designed by work package 2 provides the blueprint for the DARE software components and their interactions demonstrated by this deliverable.

The execution mapping services is informed by and provides input for the data lineage services in WP3. The DARE API created in WP3 provides an abstract layer for communication with the execution platform via the mapping services.

Work package 5 provides the testbed infrastructure with the Kubernetes[1] cluster that hosts the API, data services, user management, provenance services and the workflow catalogue which all interact with the mapping services.

The design and implementation of the execution mapping services is driven by the priorities and needs posed by the EPOS[2] and IS-ENES[3] use cases in work packages 6 and 7. In collaboration with WP8, the execution mapping services were demonstrated in training events for each of the use case communities of WP6 and WP7.

1.3 Methodology and Structure of the Deliverable

We describe the improvements to the abstract specification services in section 2. The integration in the DARE platform is described in section 3. A summary can be found in section 4.

[1] https://kubernetes.io/

[2] EPOS European Plate Observing System https://www.epos-eu.org/

[3] IS-ENES InfraStructure for the European Network for Earth System Modelling https://www.wur.nl/en/show/IS-ENES-InfraStructure-for-the-European-Network-for-Earth-System-Modelling.htm

2 DARE Abstraction Specification and Execution Mapping Services

2.1 Abstraction Specification Components

The data-driven abstraction specification DARE software components are produced by the extension and improvement of dispel4py. These components will allow for the context-agnostic, abstract specifications of methods addressing data, computing and complexity extremes.

2.1.1 CWL

The **Common Workflow Language** (CWL)[4] orchestrates the top-level workflows and provides an abstraction for task-based workflows across multiple platforms. The CWL W3C proposal is widely used by research communities as an abstraction of workflow descriptions. It enables portability and reproducibility. CWL comes with rich support for the production of provenance traces.

In existing implementations of the DARE use case scenarios, the workflow steps were combined in a bash script with poor portability between platforms. To address this issue, CWL is used for abstract specifications by WP6 and WP7 to organise execution of ensembles of computationally demanding simulation code. These tools will make it easier for DARE to interoperate and integrate with other communities in addition to the DARE user communities. The provenance gathering has been extended to include provenance output from CWL seamlessly.

2.1.2 dispel4py

DARE dispel4py[5] is responsible for the abstraction within each task of a CWL workflow to provide a mapping to a parallel platform, such as a mapping to Message Passing Interface (MPI) for distributed memory clusters, and a mapping taking advantage of Python multiprocessing for shared memory systems. The **dispel4py** workflow system scales to large environments by exploiting *fine-grained* data parallelism.

dispel4py runs in containers managed by Kubernetes and controlled using CWL as well as in directly controlled contexts, such as a user's laptop. CWL manages file-processing tasks. In contrast, **dispel4py** works at a more fine-grained level. Its main concept is data streaming and computation is described using data processing elements (PE) that are coordinated by the data flowing as ordered streams of data units between them. Each PE can consume data-units from each input stream and generate data units on each output stream at rates determined by its embedded logic, e.g., a transformation may be applied to each unit resulting in an output, whereas a sum would need to consume all input units before emitting a single output data unit. A rolling average, would consume the set span of samples before emitting its stream of data units.

2.1.3 PE Catalogue

The specified DARE software components and workflows are stored in the **dispel4py** or CWL registry. The registry stores complete workflows as well as individual PEs and their descriptions. These can be imported by third parties as and when required. The registry is collecting a rich library of PEs produced by the implementations and extensions of the use cases. The implementation[6] was originally developed as part of the VERCE project[7] [Atkinson *et al.* 2015].

[4] https://www.commonwl.org

[5] https://gitlab.com/project-dare/dispel4py/tree/master

[6] https://gitlab.com/project-dare/d4py-registry

[7] http://www.verce.eu

2.1.4 Extension of the PE library

There are three groups of PE library extensions i.e., general-purpose algorithms, and two specialised collections serving the needs of seismologists in EPOS and the needs of climate modelers in IS-ENES.

General PEs: These PEs are not domain specific and will be applied in many scenarios:

- Match or Join: this PE pairs up input data from two input streams, matching data items according to a join condition

For the EPOS rapid assessment (RA) use case the PE library was extended with several new processing element implementations:

- **ReadStream**: Create a stream of real and synthetic input data from miniseed formatted files with matching time windows
- **Norm**: a preprocessing step that calculates the norm of the input streams
- Calculate **peak ground motion**
- **Visualisation:** plot maps to visualise the peak ground motion parameters
- Produce **GeoJSON** to enable integration with other geo-referencing tools

RA makes use of the Match PE: after the preprocessing pipeline of the synthetic and the real input datasets, this PE pairs up real and synthetic traces from the same station and with the same time window.

For the IS-ENES climate use case the following PE was created:

- **icclim** function: this PE applies the icclim[8] function **icclim.indice()** which is parameterised at the creation of the workflow.

2.2 Execution Mapping Components

The execution mapping interprets the abstract data flow created by a user to create a concrete enactment environment and to orchestrate the data processing.

2.2.1 Kubernetes

Kubernetes provides and manages the enactment platform, provided by WP5 (See §3.1). It also orchestrates the MPI execution and, in the future, the Apache Spark execution.

2.2.2 SPECFEM3D

The software package **SPECFEM3D Cartesian**[9] simulates seismic wave propagation at the local or regional scale and performs full waveform imaging (FWI) or adjoint tomography based upon the spectral-element method (SEM). It is a parallel application for HPC environments with distributed memory and MPI. To make this application available in the DARE cloud environment, a docker[10] **image** and a **docker-compose** configuration was created to create a cluster with MPI and SPECFEM3D. This docker configuration provides the complete environment for deploying SPECFEM3D to the DARE testbed. Coupled with the DARE API this component performs simulations in a Kubernetes distributed compute cluster on demand. The synthetic data created in these simulations form the input for the seismology rapid assessment use case.

2.2.3 Docker

A docker file is provided for each of the use cases (WP6&7) to deliver a standard and familiar environment for domain users combined with the **dispel4py** client toolkit. In the case of WP6

[8] https://icclim.readthedocs.io/en/latest/

[9] https://specfem3d.readthedocs.io/en/latest/

[10] https://www.docker.com/

(seismology) this image contains the latest *obspy* release[11], for WP7 (climatology) the *icclim* toolkit[12] is included.
dispel4py MPI Docker containers are deployed to Kubernetes to create MPI clusters on demand.

2.2.4 dispel4py mappings

This section describes the new features that were added to the dispel4py execution engine to support the use case requirements. These have been developed substantially, to include dynamic optimisation.

2.2.4.1 Mappings

A prototype mapping for dispel4py explores the feasibility of **Kafka**[13] as the distributed queue engine in combination with Docker containers that wrap PEs and their dependencies. The Kafka mapping is suitable for distributed memory environments (e.g. High Performance Computing (HPC) clusters) and represents an alternative to the MPI mapping. Since Kafka has been designed to take into account the needs of data-streaming applications (e.g. scalability, reliability), our aim is to create dispel4py workflows using Kafka query engine and compare the performance with the MPI mapping using the Kubernetes infrastructure.

2.2.4.2 Dynamic optimisation of dispel4py

A new dynamic optimisation mechanism has been developed and tested on a variety of workflows. This was reported at this year's WORKS conference [Liang *et al.* 2020]. This provides new optimisation where the deployment and replication of PEs is increased in response to load, thereby developing the potential to respond to data-dependent loads and variations in platform performance. The data-streams exploit a different service, mqz, and the adaptive algorithms are parameterised from previous monitoring runs of the same workflow. In due course, those parameters may be mined from provenance records.

The relevant open-source version of **dispel4py** is available in the DARE code repository in a new experimental branch: https://gitlab.com/project-dare/dispel4py/-/tree/experimental. It provides two new mappings[14] for the dynamic deployment of dispel4py workflows: using zmq library (zmq_multi.py) and the multiprocessing library (dynamic.py). These deliver two optimisation techniques[15] for the static deployment of dispel4py workflows: Naïve Assignment (naive_partition.py), and Staging (stage.py). These have been tested and validated by optimisation experiments conducted in conjunction with WP6[16]. These experiments use the Rapid Assessment (RA) workflow, combining all of the stages, a seismic cross-correlation workflow, legacy of the VERCE project, and standard examples from astrophysics and window-joins. Readers are advised to read detailed descriptions[17] before using these new dispel4py mechanisms. These notes contain three sections: (1) the current "mini-workflows" for the RA use case, (2) the corresponding integrated workflows (one per phase) for the RA use case, and (3) the semi-stream workflows for the RA use case, used in the optimisation experiments.

[11] https://docs.obspy.org/index.html

[12] https://icclim.readthedocs.io/en/latest/

[13] https://kafka.apache.org/

[14] https://gitlab.com/project-dare/dispel4py/-/tree/experimental/dispel4py/new

[15] https://gitlab.com/project-dare/dispel4py/-/tree/experimental/dispel4py/optimization

[16] https://gitlab.com/project-dare/WP6_EPOS/-/tree/optimization_experiments/processing_elements/OPTIMIZATIONS_EXPERIMENTS

[17] https://gitlab.com/project-dare/WP6_EPOS/-/blob/optimization_experiments/processing_elements/OPTIMIZATIONS_EXPERIMENTS/README.md

These experiments revealed issues with the current RA workflows. The first mini-workflow (create_download_json.py) and the last PE of the last mini-workflow (dispel4pyRAMapping.py) have an incompatibility with the Multiprocessing library: create_download_json.py uses requests.get for downloading the synthetic data that is incompatible with the Multiprocessing library, and dispel4pyRAMapping.py uses plt.subplots that is not compatible with the Multiprocessing library. The work-around is the 3rd version (semi-stream workflows. The total stream workflows can be run using the simple/sequential mapping.

2.2.5 Wider support of CWL

CWL is being used as an intermediary language between the client specification of a dataflow and the service executing the dataflow. A first prototype shows how a **dispel4py** workflow can be translated to CWL, opening the door to interoperability with other workflow engines that support CWL. Adding a CWL interface to **dispel4py** supports clients that use CWL to describe their workflows. Combining this with **dispel4py** streaming features creates a rich environment for data intensive use cases.

2.2.5.1 Provenance

dispel4py with provenance automatically generates the provenance wrappers for a **dispel4py** workflow, specified by the user as a command line parameter when executing **dispel4py** on the target platform.

2.3 Execution API

The DARE Execution API enables the distributed and scalable execution of DARE components via a HTTP interface that provides access to DARE execution services in a language-independent fashion, thereby separating the client environment from the DARE execution platform. The relevant source code and documentation can be accessed via the project's GitLab repository[18].

The current release of the DARE API supports the execution and monitoring of distributed SPECFEM3D simulations and dispel4py workflows on a Kubernetes cluster that is created and removed on-demand. The API implementation is designed to extend to other execution contexts in later releases.

Through the role-based access, the control execution API accesses the Kubernetes API to spawn container clusters on demand, while at the same time enabling shared file system access with itself and the execution contexts and monitoring the status of executed jobs. All execution contexts are built to use Common Workflow Language (CWL) which allows for dynamically parameterized executions.

In addition, the execution API offers services such as uploading/downloading and referencing of data and process monitoring.

A brief overview of the API's functionality can be found at the relevant wiki page of the project[19].

2.4 Data services

The data referencing as well as the uploading and downloading data for stage-in or stage-out in the context of a workflow execution are available within the services of the DARE platform as a shared file system. In addition, the workflows can use the EUDAT B2DROP[20] service for sharing data.

In this phase, we show how this service is being exploited for the climate use case. It is planned to be used in the next phase also for the EPOS use case.

[18] https://gitlab.com/project-dare/dare-api

[19] https://gitlab.com/project-dare/dare-api/wikis/Execution-API-brief-documentation

[20] https://eudat.eu/services/b2drop

3 Summary

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 method's semantics unchanged. In the future we will start the integration of an EOSC-Friendly AAI for the DARE platform, create improvements of the semantic catalogues and the DARE Knowledge Base (DKB), along with the DARE workflow optimiser.

In this iteration, important new features were added to the dispel4py library and the execution mappings to support the DARE use cases and enable necessary abstractions for the next stage, in which we will enrich the execution mapping services to optimise workflow executions. The optimiser will use the provenance services to assess past performance; annotated PE descriptions in the catalogue will inform deployment and distribution of workflows and components; extensions to the DARE API will be integrated by the mapper.

4 Bibliography

[Atkinson *et al.* 2015] Atkinson M.P., Carpené M., Casarotti E., Claus S., Filgueira, R., Frank A., Galea M., Garth T., Gemünd A., Igel H., Klampanos I.A., Krause A., Krischer L., Leong S.H., Magnoni F., Matser J., Michelini A., Rietbrock A., Schwichtenberg H., Spinuso A., and Vilotte J-P., *VERCE delivers a productive e-Science environment for seismology research*, in Proceedings of IEEE eScience 2015.

[Liang *et al*. 2020] Liang Liang, Filgueria Rosa and Yan Yan, *Adaptive Optimizations for Stream-based Workflows*, in proceedings WORKS 2020, Oct. 2020.