H2020-EINFRA-2017
EINFRA-21-2017 - Platform-driven e-infrastructure innovation
DARE [777413] "Delivering Agile Research Excellence on European e-Infrastructures"

Data-driven Abstraction Specification and Execution Mapping Services Toolkit I

Project Reference No	777413 — DARE — H2020-EINFRA-2017 / EINFRA-21-2017
Deliverable	D4.3 Data-driven Abstraction Specification and Execution Mapping Services Toolkit I
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	30/06/2019 Deadline extended to 31/07/2019 in agreement with PO
Submission Date	31/07/2019
Status	Final Draft
Editor(s)	Amy Krause (UEDIN)
Contributor(s)	Alessandro Spinuso (KNMI), Thanasis Davettas, Antonis Koukourikos (NCSR-D)
Reviewer(s)	Andreas Oikonomopoulos (NCSR-D)
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

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
0.1	24/06/2019	ToC	NCSR-D
0.2	09/07/2019	First draft	UEDIN
0.5	22/07/2019	Execution API info	NCSR-D
0.8	25/07/2019	Final draft	UEDIN
0.9	26/07/2019	Internal Review	NCSR-D
1.0	29/07/2019	Review comments addressed and document finalised	NCSR-D

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. These components allow for the context-agnostic, abstract specifications of methods addressing data, computing and complexity extremes. New features were added to dispel4py to extends the support for distributed mappings and more functionality as processing element implementations.
The abstractions defined by dispel4py are exposed through the DARE API. The DARE API is the entry point of DARE that among other allows the access to the computational resources
of the platform.

Table of Contents

1 Introduction 5

1.1 Purpose and Scope 5

1.2 Approach and relationship with other Work Packages and Deliverables 5

1.3 Methodology and Structure of the Deliverable 5

2 DARE Abstraction Specification and Execution Mapping Services 6

2.1 Abstraction Specification Components 6

2.1.1 CWL 6

2.1.2 dispel4py 6

2.1.3 PE Catalogue 6

2.1.4 Extension of the PE library 6

2.2 Execution Mapping Components 7

2.2.1 Kubernetes 7

2.2.2 SPECFEM3D 7

2.2.3 Docker 7

2.2.4 dispel4py mappings 7

2.2.4.1 Mappings 7

2.2.4.2 Provenance 8

2.2.4.3 Python 3 8

3 Integration in DARE Platform 9

3.1 Execution API 9

3.2 Data services 9

4 Summary and Next Steps 10

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. 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 described in 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 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 and IS-ENES 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 and an outline of further steps for upcoming releases can be found in section 4.

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)[1] orchestrates the top-level workflows and provides an abstraction for task-based workflows across multiple platforms. CWL is widely used in e-science communities as the abstraction of workflow descriptions enables portability and reproducibility. CWL comes with rich support for the collection of provenance.
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. These tools will make it easier for DARE to interoperate and integrate with other communities in addition to the DARE user communities.

2.1.2 dispel4py

DARE dispel4py[2] 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 MPI for distributed memory clusters, and a mapping taking advantage of Python multiprocessing for shared memory systems. Dispel4py scales to large environments by using data parallelism.
In collaboration with the top-level orchestration by CWL, dispel4py works at a more fine-grained level. Its main concept is data streaming and tasks are described as data processing elements that are coordinated by the data flowing between them.

2.1.3 PE Catalogue

The data-driven abstraction specification DARE software components are stored in the dispel4py **catalogue**, or 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[3] was originally developed as part of the VERCE project.

2.1.4 Extension of the PE library

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

[1] https://www.commonwl.org
[2] https://gitlab.com/project-dare/dispel4py/tree/master
[3] https://gitlab.com/project-dare/d4py-registry

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[4] function **icclim.indice()** which is parameterised at the creation of the workflow.

2.2 Execution Mapping Components

The execution mapping interprets the abstract dataflow 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. It also orchestrates the MPI execution and, in the future, the Apache Spark execution.

2.2.2 SPECFEM3D

The software package **SPECFEM3D Cartesian** 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 **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 provide a familiar environment for domain users and combine this with the **dispel4py** client toolkit. In the case of WP6 (seismology) this image contains the latest *obspy* release, for WP7 (climatology) the *icclim* toolkit 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.

2.2.4.1 Mappings

A prototype mapping for dispel4py explores the feasibility of **Kafka** 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. 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.
CWL is being used 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.

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

2.2.4.2 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.

2.2.4.3 Python 3

A new version of dispel4py was released to provides support for Python 3, which is the version that is used throughout the DARE project.

3 Integration in DARE Platform

3.1 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[5].

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[6].

3.2 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 filesystem. In addition, the workflows can use the EUDAT B2DROP 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.

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

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

4 Summary and Next Steps

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.