

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

Dr. Rosa Filgueira, University of Edinburgh, EPCC



Delivering Agile Research Excellence on European eInfrastructures

Aim: To empower domain experts to invent and improve their methods and models

How: By providing a new platform and a working environment

Outcome: Tools/frameworks/APIs for data-driven experiments and rapid prototyping

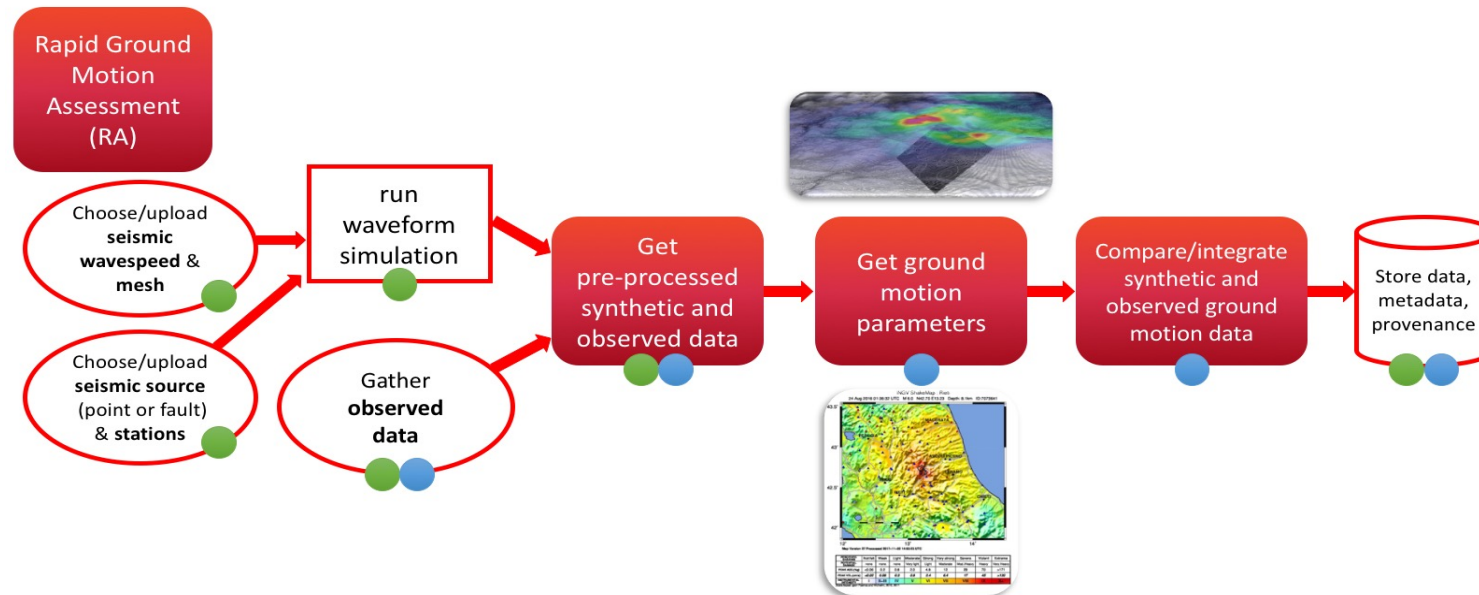
Domains: Seismology (EPOS) and Climate (IS/ENES2)

- * Seismology: EPOS
- * Climate: IS/ENES2

Rapid Ground Motion Assessment (RA)

First seismology use case:

- * Quickly analyse earthquakes
- * Model the ground motion after earthquakes
- * Rapid assessment of earthquakes' impact, and emergency response



RA – Summary Steps (I)

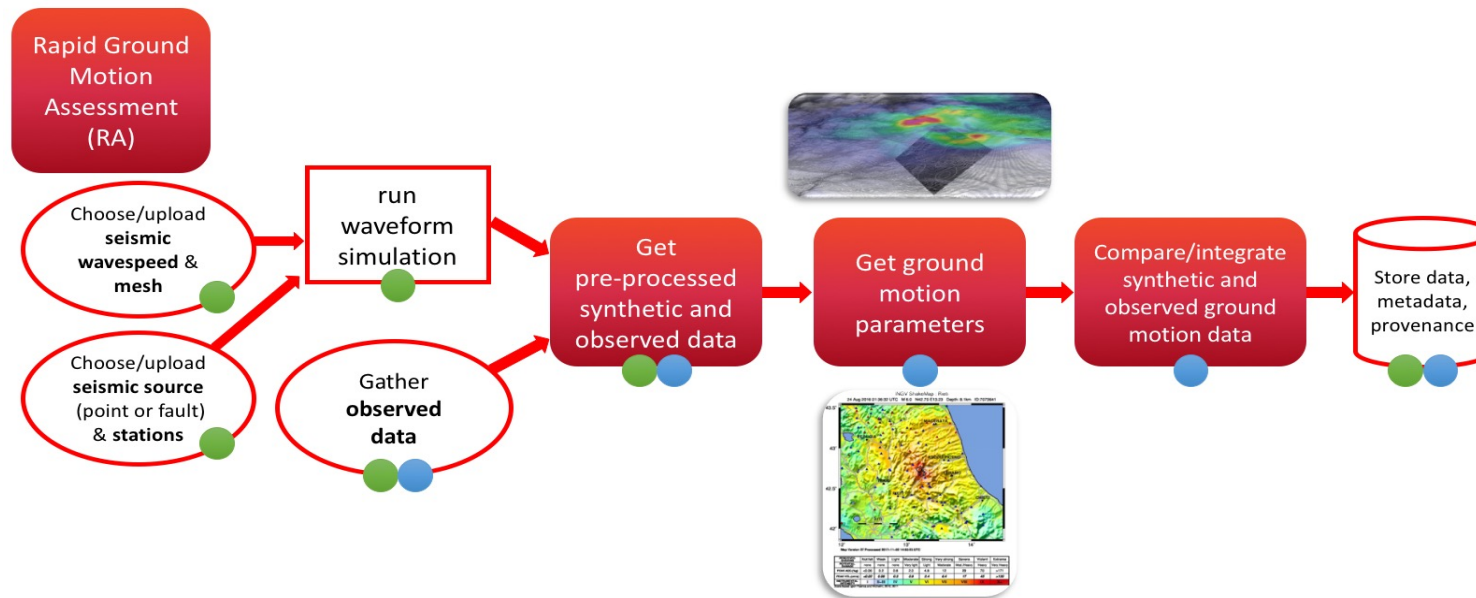
1. Dockerize Specfem3D

Build a CWL workflow for generating synthetic data

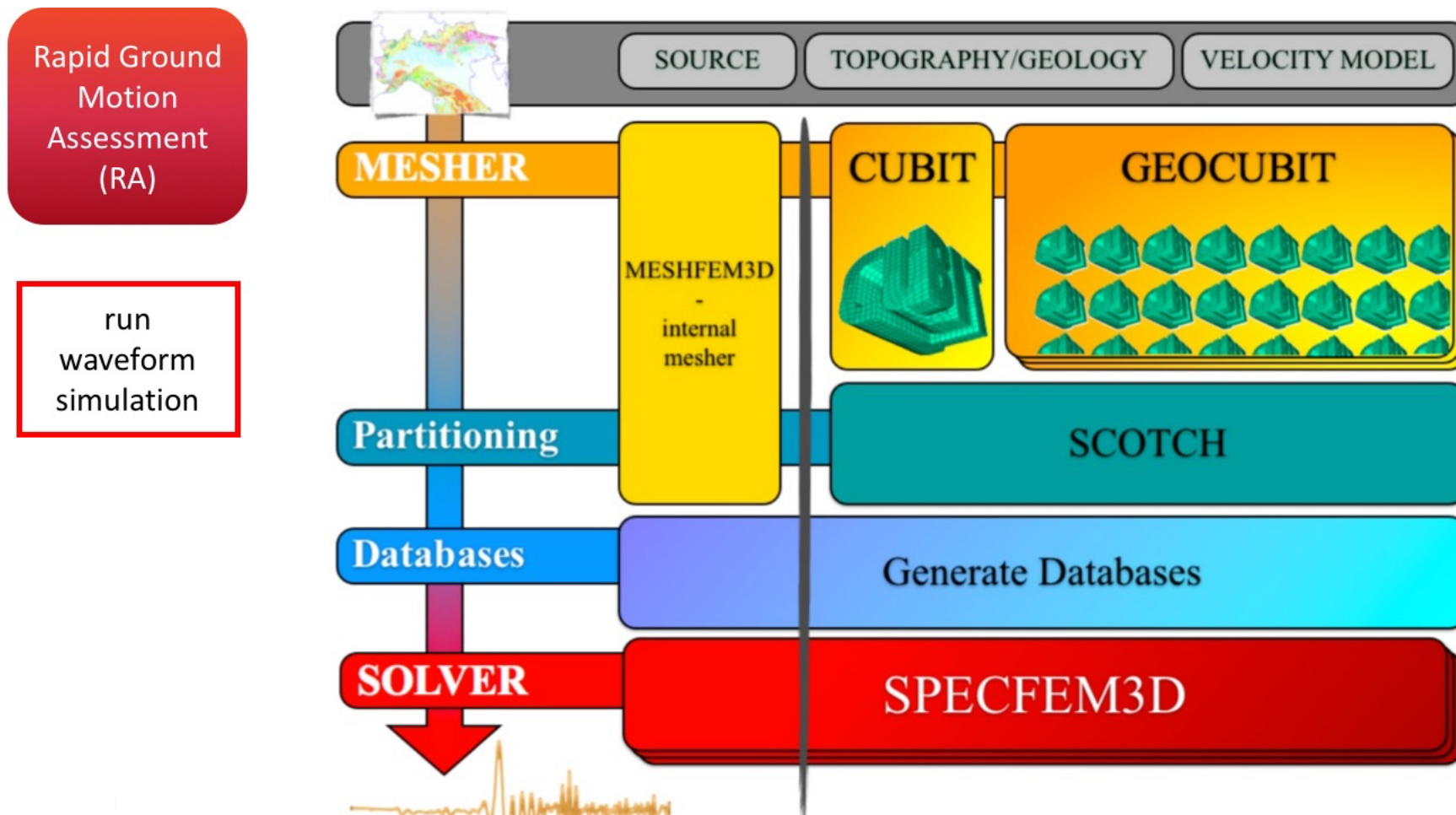
2. Build dispel4py workflows to represent each part of the RA (**)

(**) Except for the generation of the synthetic data

3. Use CWL to connect RA dispel4py workflows



Seismic Waveform Simulation: Specfem3D + MPI cluster

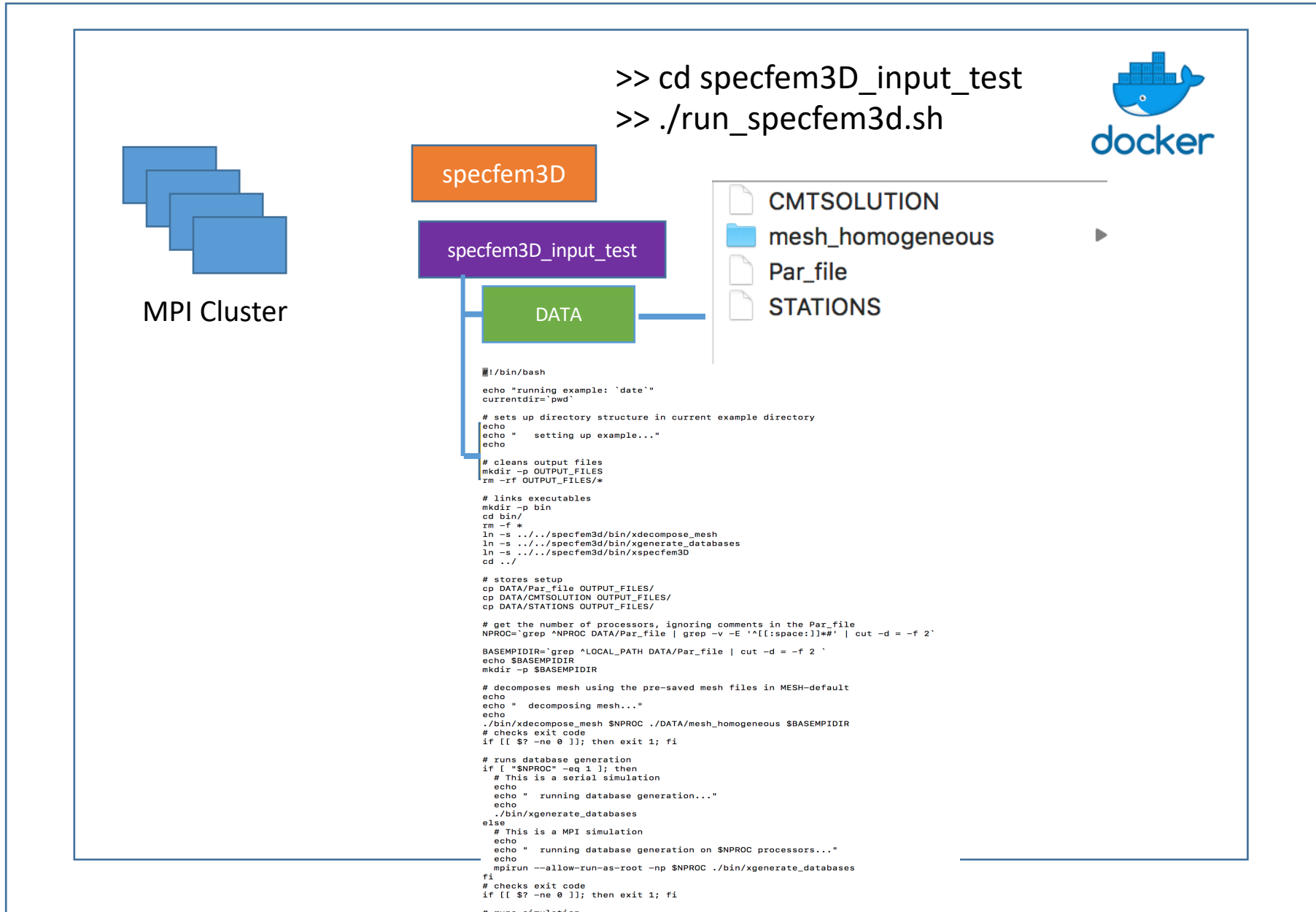


Seismic Waveform Simulation: Specfem3D + MPI cluster

We run it before
In SuperMUC -
HPC-cluster -
256 cores

Test Case: RA

run waveform
simulation



Open standard for describing

- workflows and tools
- platform-independent

} consistent way to
connect programs

1st-tool.cwl

```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0
class: CommandLineTool
baseCommand: echo
inputs:
  message:
    type: string
    inputBinding:
      position: 1
outputs: []
```

```
$ cwl-runner 1st-tool.cwl echo-job.yml
[job 1st-tool.cwl] /tmp/tmpmM5S_1$ echo \
  'Hello world!'
Hello world!
[job 1st-tool.cwl] completed success
{}
Final process status is success
```

Next, create a file called `echo-job.yml`, containing the following boxed text, which will describe the input of a run:

echo-job.yml

```
message: Hello world!
```

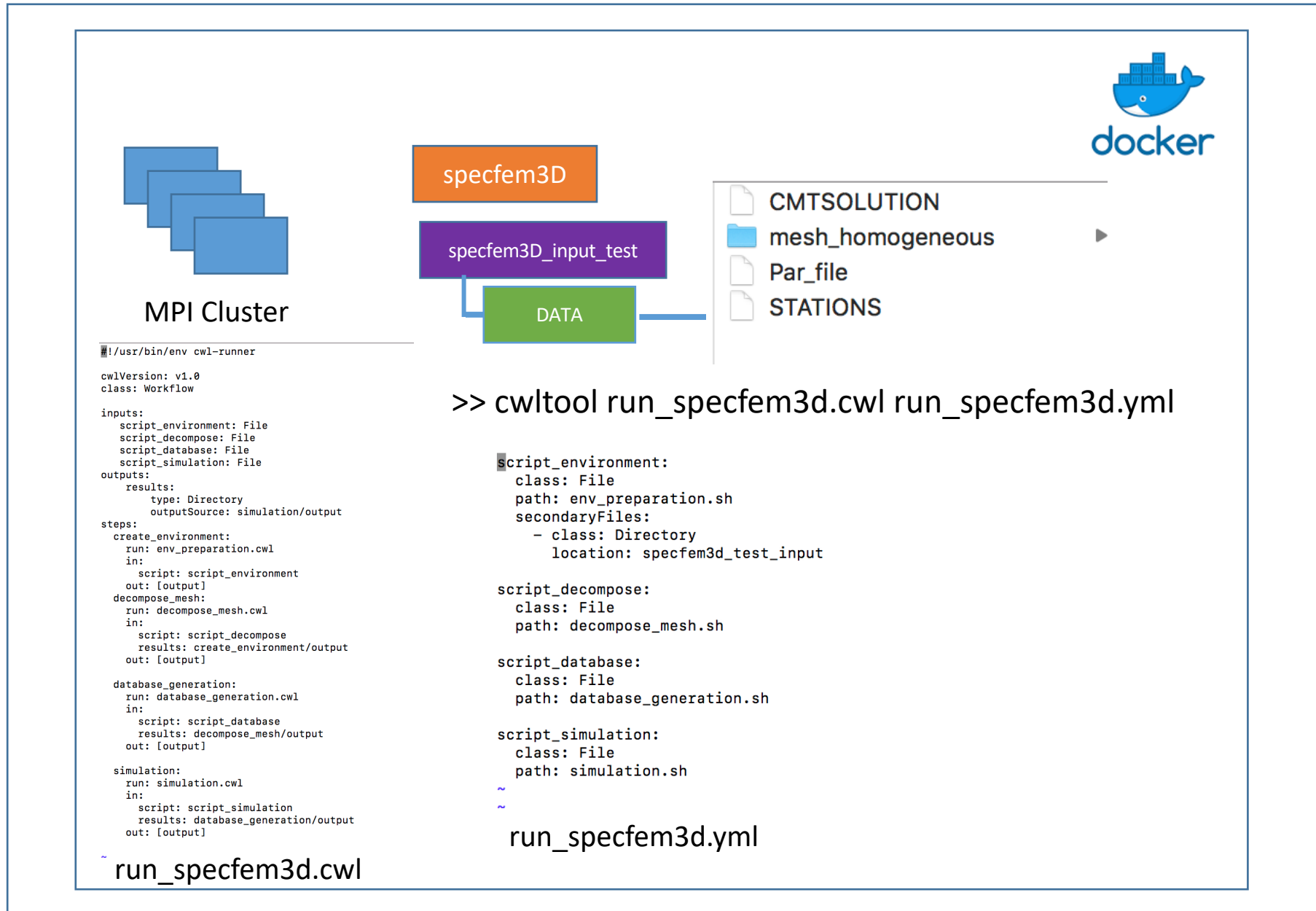
Rules to describe each
command line tool and
its parameters

Seismic Waveform Simulation: Specfem3D + MPI cluster + CWL

We run it before
In SuperMUC -
HPC-cluster -
256 cores

Test Case: RA

run waveform
simulation



RA – Summary Steps (I)

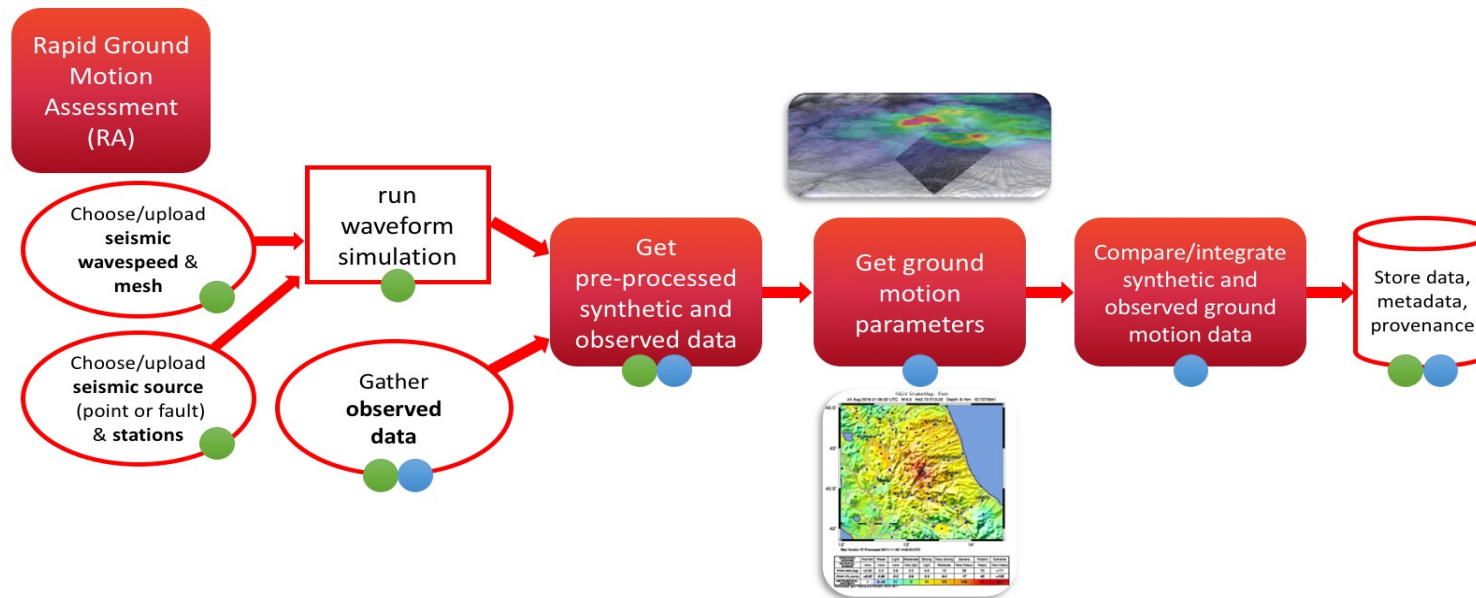
1. Dockerize Specfem3D

Build a CWL workflow for generating synthetic data

2. Build dispel4py workflows to represent each part of the RA (**)

(**) Except for the generation of the synthetic data

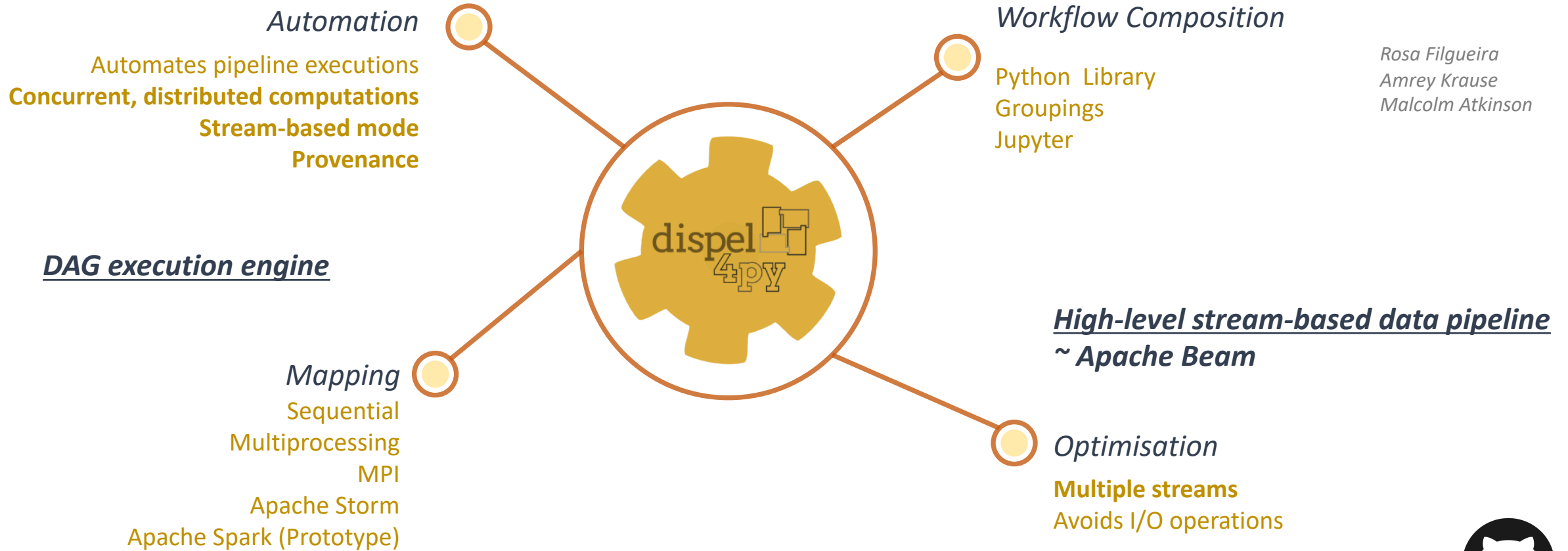
3. Use CWL to connect RA dispel4py workflows



dispel4py parallel stream-based dataflow system

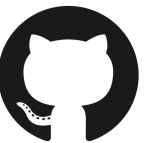


Rosa Filgueira
Amrey Krause
Malcolm Atkinson



Key-features: Automatic mappings to different engines, concurrent & stream-based
Embarrassing parallel data-intensive applications

<https://github.com/dispel4py/dispel4py>



Graph

- Connections among PES
- Abstract workflow

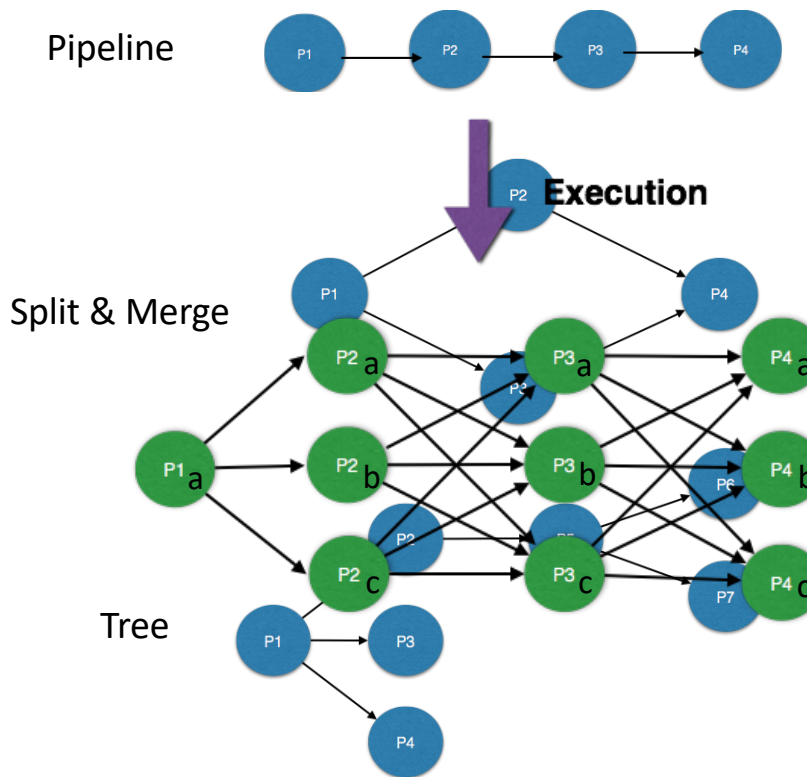
Instance

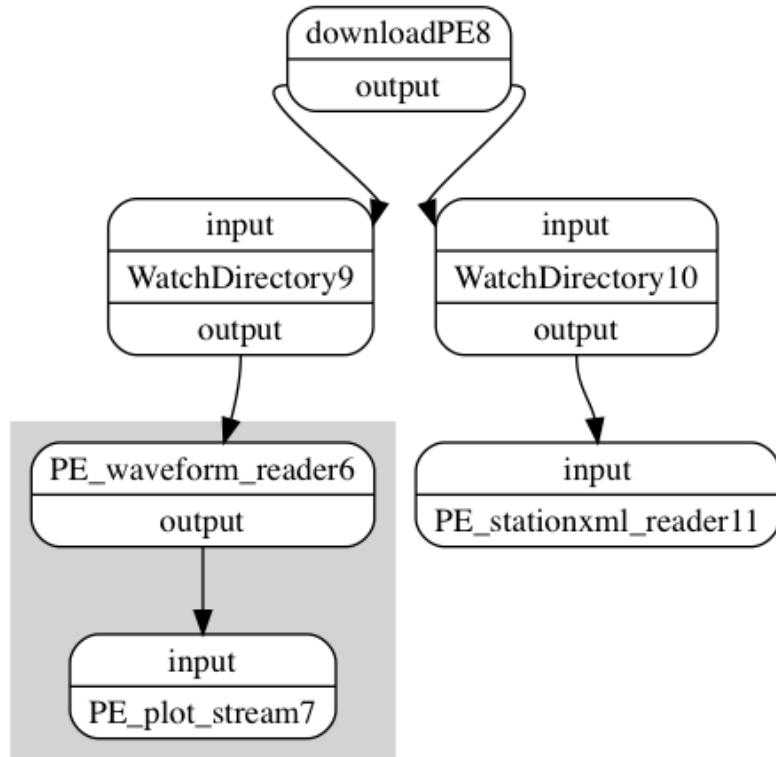
- Each PE is translated into one or more instances in run-time
- Each instance runs in a process
- dispel4py does it for you
- Concrete workflow

Mappings

- Sequential, multiprocessing, MPI

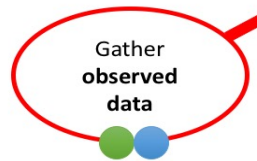
+ Example of graphs

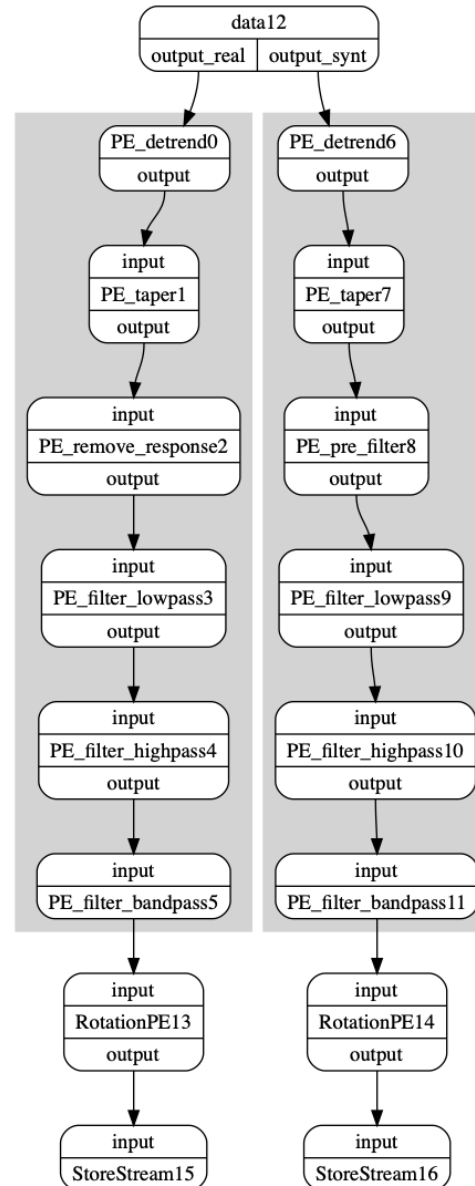




Given a list of waveforms simulated with SPECSEM3D for a specific earthquake, the workflow downloads real waveforms corresponding to the same earthquake.

Test Case: RA



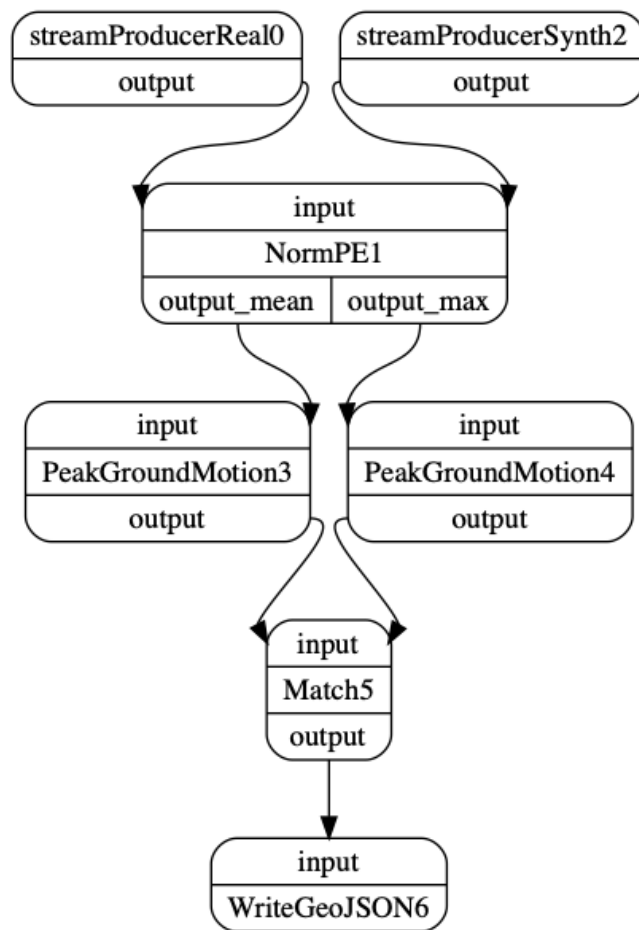


Similar preprocessing steps in synthetic and observed data

Assures the consistency between data and synthetics

Test Case: RA

Get pre-processed synt and data



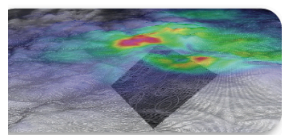
Ground motion parameters:

Peak ground values of displacement, velocity and acceleration.

Two types of normalisation – Mean & Max

Two set of PGM outputs – Max & Mean

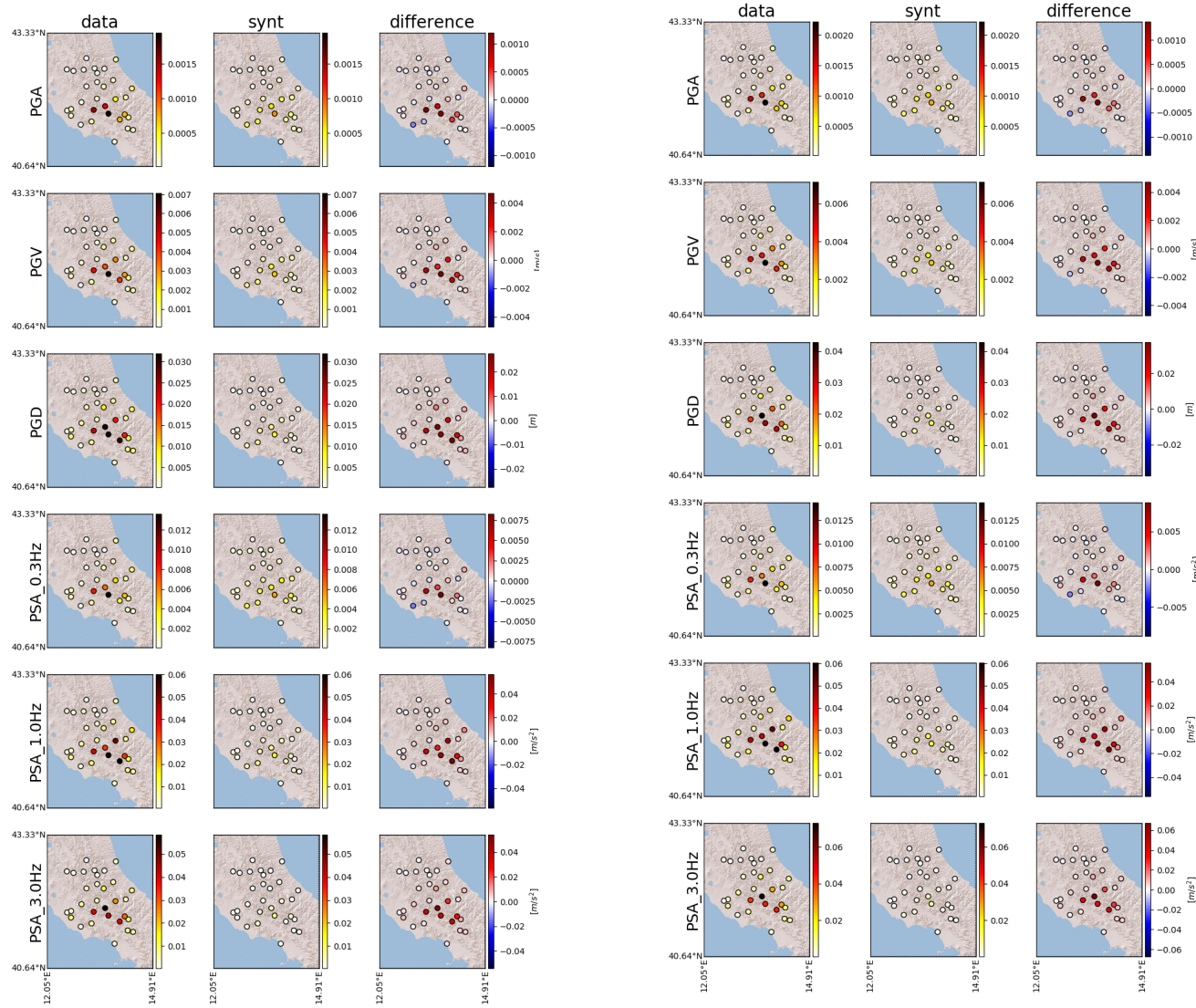
Test Case: RA



Get ground motion parameters

Compare/integrate synthetic and observed ground motion data

RA – Ground motion parameters maps



Waveform propagation snapshots and maps of ground motion parameters are fundamental for a visual representation of the earthquake

RA – Summary Steps (I)

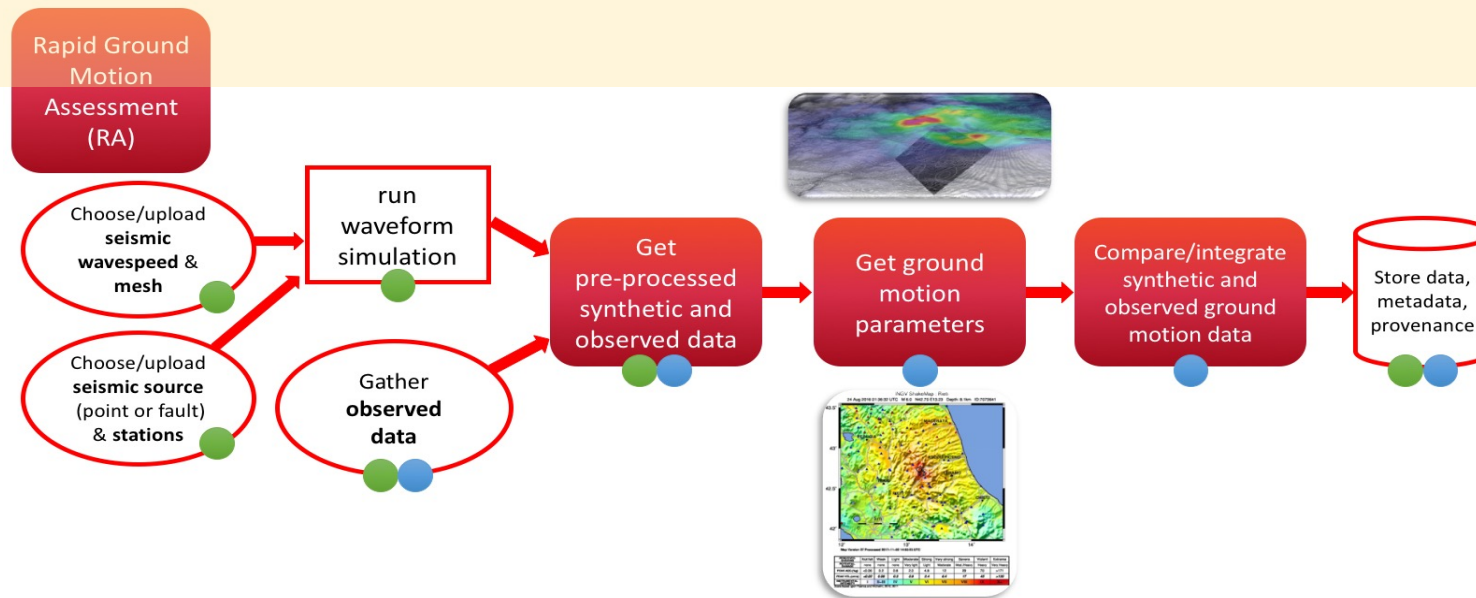
1. Dockerize Specfem3D

Build a CWL workflow for generating synthetic data

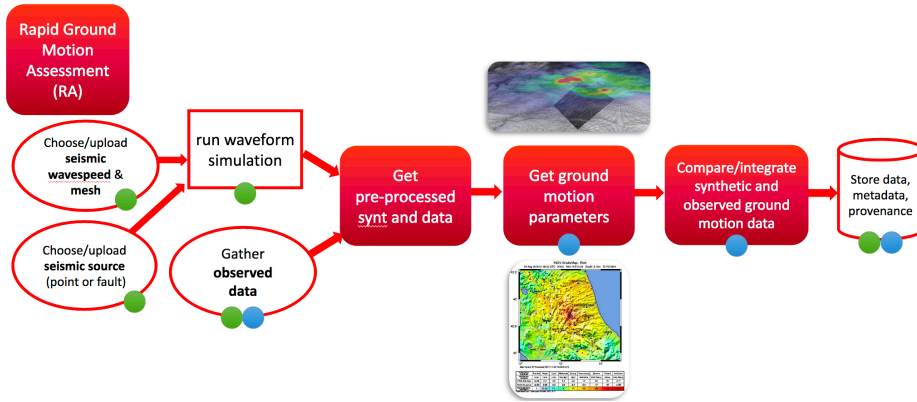
2. Build dispel4py workflows to represent each part of the RA (**)

(**) Except for the generation of the synthetic data

3. Use CWL to connect RA dispel4py workflows



RA – dispel4py + CWL



dispel4py +  COMMON WORKFLOW LANGUAGE → semantics and descriptions

```
>> cwltool --provenance run-ra/ --full-name "Rosa Filgueira"
run_ra.cwl run_ra.yml
```

```
>> cwlprov --directory run-ra/ validate
Valid CWLProv RO: run-ra
```

```
>> cd run-ra/
>> cwlprov info
Research Object of CWL workflow run
Research Object ID: arcp://uuid,7b810637-40bb-467a-9c0a-a89865bf3c09/
Profile: https://w3id.org/cwl/prov/0.6.0
Workflow run ID: urn:uuid:7b810637-40bb-467a-9c0a-a89865bf3c09
Packaged: 2019-09-17
```

```
>> cwlprov who
Packaged By: cwltool 1.0.20190228155703 <urn:uuid:bbfe3471-2b87-4b11-975a-b96b2f24f4f8>
Executed By: Rosa Filgueira <urn:uuid:7d32cf62-d9c8-467e-8f58-5a74b70091e5>
```

```
#!/usr/bin/env cwl-runner
```

```
cwlVersion: v1.0
class: Workflow
```

```
inputs:
```

```
create_env_script: File
download_workflow: File
download_argument_f: File
preprocess_workflow: File
ra_workflow: File
ra_argument_d: string
```

```
outputs:
```

```
misfit_data:
  type: Directory
  outputSource: preprocess_data/output
pgm_data:
  type: Directory
  outputSource: rapid_assessment/output
```

```
steps:
```

```
create_env:
  run: create_env.cwl
  in:
    script: create_env_script
  out: [output]
download_data:
  run: dispel4py_download.cwl
  in:
    workflow: download_workflow
    argument_f: download_argument_f
    misfit_data: create_env/output
  out: [output]
preprocess_data:
  run: dispel4py_preprocess.cwl
  in:
    workflow: preprocess_workflow
    misfit_data: download_data/output
  out: [output]
rapid_assessment:
  run: dispel4py-RA-pgm_story.cwl
  in:
    workflow: ra_workflow
    argument_d: ra_argument_d
    misfit_data: preprocess_data/output
  out: [output]
```

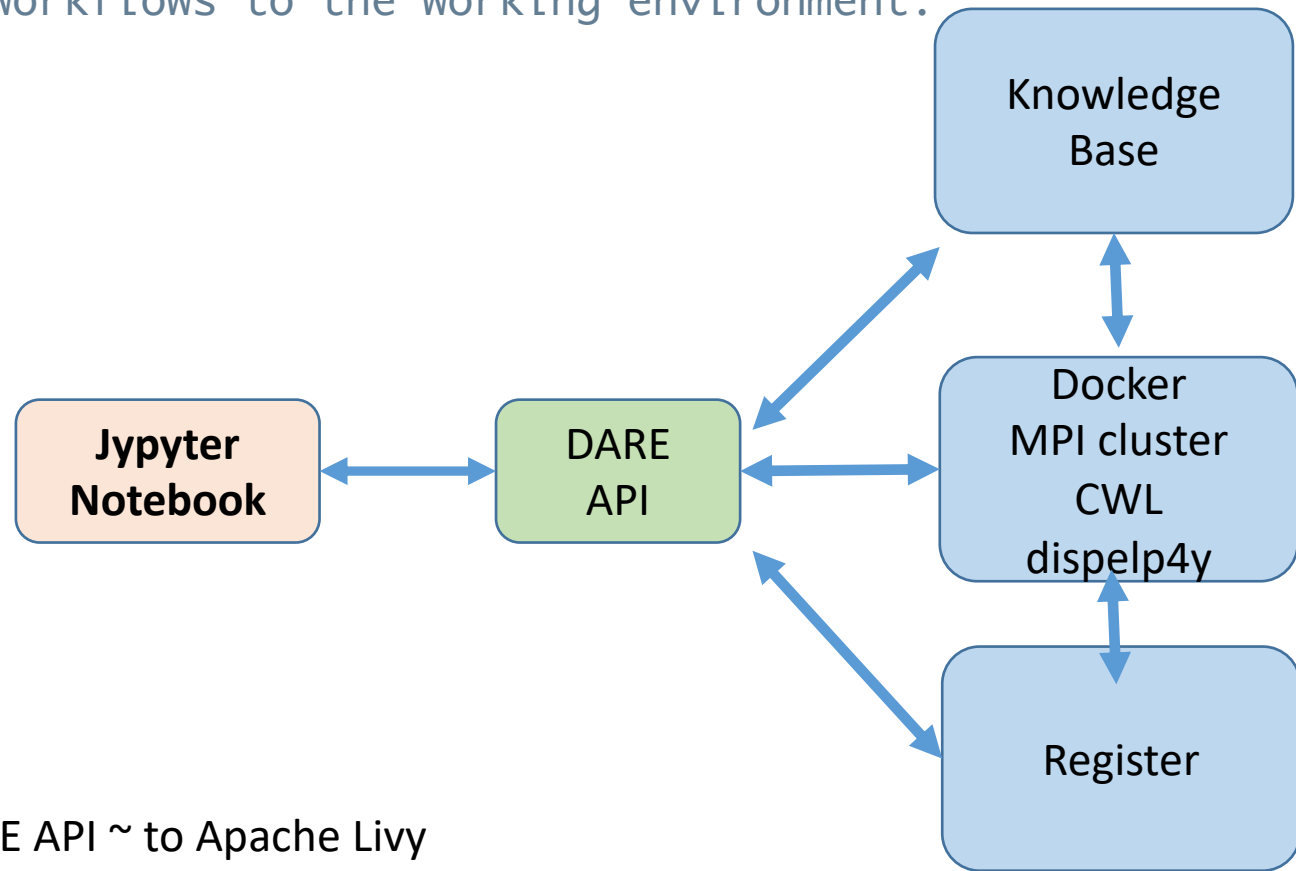
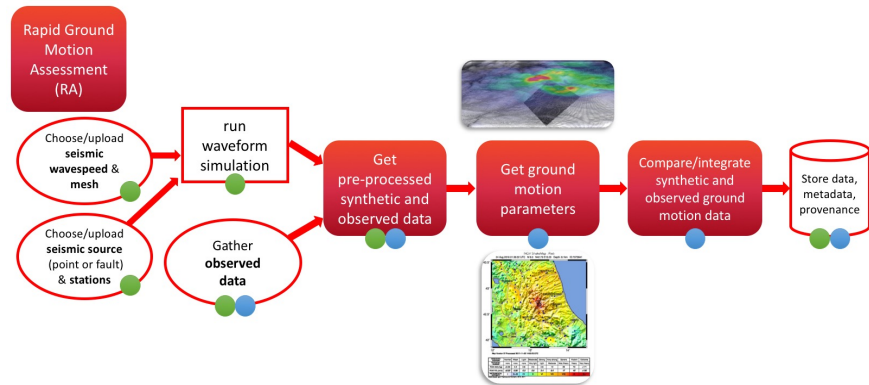
RA – Summary Steps (II)

4. DARE API – Workflows as a Services

5. Infrastructure orchestrations using Kubernetes: MPI cluster, dispel4py, CWL, SPECFEM3D

6. Jupyter Notebooks to applications/workflows to the working environment:

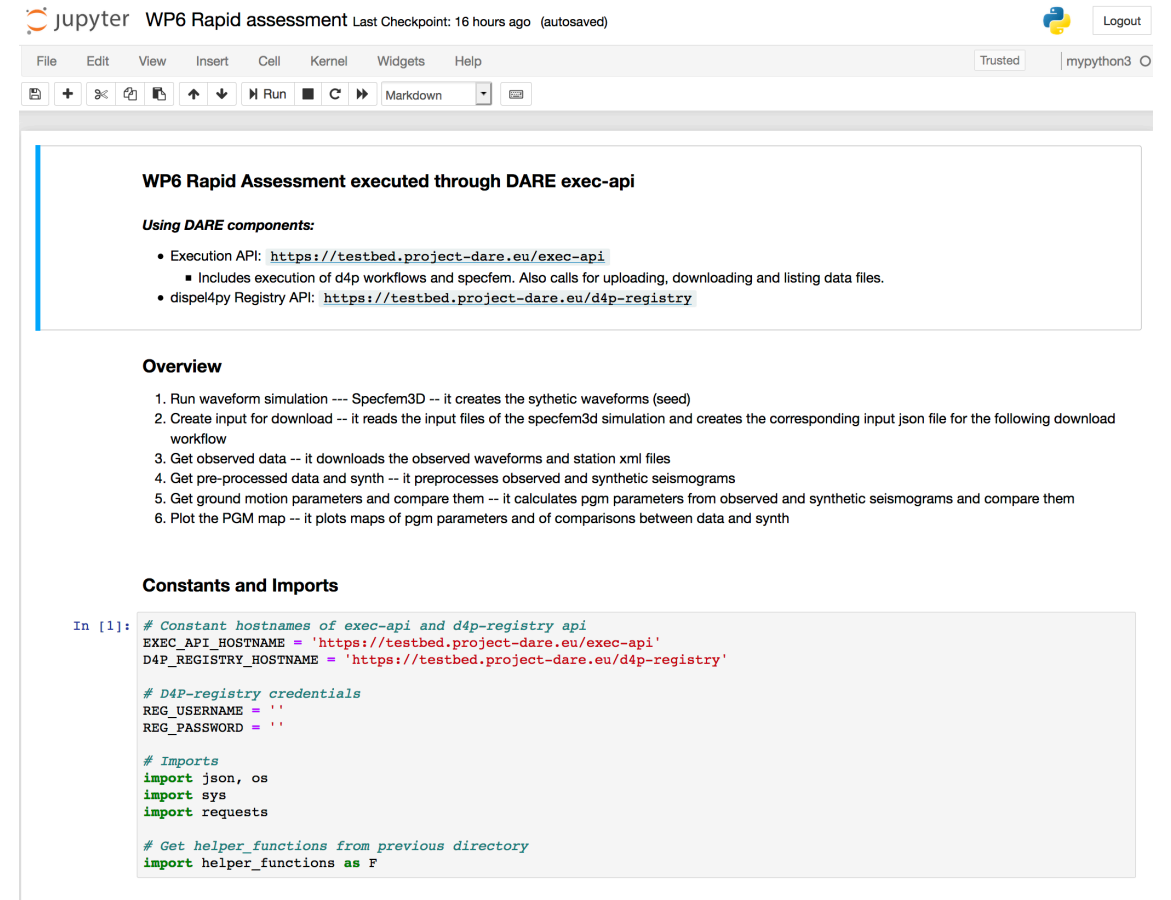
* Talk with the DARE API



*DARE API ~ to Apache Livy

DARE API

- Web service
- Acts as an intermediary between:
 - users' applications
 - the underlying computing resources
- Provisions a computing environment
 - Docker MPI cluster spawned on demand / Kubernetes
- Runs and monitors an application
- Collect its provenance and results



jupyter WP6 Rapid assessment Last Checkpoint: 16 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | mypython3

WP6 Rapid Assessment executed through DARE exec-api

Using DARE components:

- Execution API: <https://testbed.project-dare.eu/exec-api>
 - Includes execution of d4p workflows and specfem. Also calls for uploading, downloading and listing data files.
- dispel4py Registry API: <https://testbed.project-dare.eu/d4p-registry>

Overview

1. Run waveform simulation --- Specfem3D -- it creates the sythetic waveforms (seed)
2. Create input for download -- it reads the input files of the specfem3d simulation and creates the corresponding input json file for the following download workflow
3. Get observed data -- it downloads the observed waveforms and station xml files
4. Get pre-processed data and synth -- it preprocesses observed and synthetic seismograms
5. Get ground motion parameters and compare them -- it calculates pgm parameters from observed and synthetic seismograms and compare them
6. Plot the PGM map -- it plots maps of pgm parameters and of comparisons between data and synth

Constants and Imports

```
In [1]: # Constant hostnames of exec-api and d4p-registry api
EXEC_API_HOSTNAME = 'https://testbed.project-dare.eu/exec-api'
D4P_REGISTRY_HOSTNAME = 'https://testbed.project-dare.eu/d4p-registry'

# D4P-registry credentials
REG_USERNAME = ''
REG_PASSWORD = ''

# Imports
import json, os
import sys
import requests

# Get helper functions from previous directory
import helper_functions as F
```

Future Work

Workflows can be optimised intelligently without the user needing to do that

- New dispel4py mappings – dynamic deployment – *ZeroMQ*

CWLProv + dispel4py Provenance – integrate different levels of provenance

Deployment of other more complex seismological test cases