DARE: Integrating solutions for Data-Intensive and Reproducible Science

Alessandro Spinuso
and the DARE team

spinuso@knmi.nl
R&D Data Technology and Observations
What’s in this talk...

- DARE Objectives and Provenance Challenges
- *Active* provenance for Data-Intensive Workflows
  (Use Case in Seismology)
- Provenance-aware Workspaces
- Conclusions
**DARE**
**Delivering Agile Research Excellence on European e-Infrastructures**

**Working environment** for professionals wrestling with challenges involving complexity of methods and data

**System & Data Engineers** $\Rightarrow$ **Research Developers** $\Rightarrow$ **Domain Expert**

- **Mapping** between abstract methods and concrete applications executed by different enactments seamlessly
- **Validation and Traceability** of runs and products: diagnose, monitor, reuse
- **Organisation of campaigns** reusing data and methods from multiple runs
- **Driven and Evaluated by communities**

Accelerate productivity of expert teams.
Lineage is DATA (Retrospective Provenance)

- Data's origins, what happens to it and where it **moves over time**
- It may include **technical metadata**: quality test results, reference values
- Ability to trace errors back to the root cause.
- Integrated in **workflow systems** to trace the data via various changes
- Its volume depends on its scope!
Data Lineage

Lineage is DATA (Retrospective Provenance)
- Data's origins, what happens to it and where it moves over time
- It may include technical metadata: quality test results, reference values
- Ability to trace errors back to the root cause.
- Integrated in workflow systems to trace the data via various changes
- Its volume depends on its scope!

Challenges in Data-Intensive workflows:

Granularity: provenance information can be too coarse or too detailed. Intermediate Data can be Materialised, as well as Volatile (streaming).

Precision: detailed capturing of data derivations in parallel operators (what input data contributed to what output).

Relevance: how to manage domain and application specific properties?

Reuse: lack in validation and understanding of the computational method => ineffective reuse of results. (Reproduce vs Reuse battle !?)
Domain properties
User’s Context
(e.g. Seismology)

Common Metadata Baseline
for Data collections
Key-features: Automatic parallelisation/mappings, concurrent & stream-based, configurable provenance

https://gitlab.com/project-dare/dispel4py
Active Provenance Capturing

Specify Workflow

Formulate Configuration types, clusters, sensors...

Apply configuration Execute

Monitor Validate

Component

Provenance Type
Active Provenance Capturing

Provenance Configuration

Specify Workflow
Formulate Configuration types, clusters, sensors...
Apply configuration
Execute
Monitor Validate

Reusable Provenance Type

Component
Provenance Type

Domain Metadata

Contextualisation Types
- SeismoType
- NetCDFType

Precision of Data Derivations

Pattern Types
- SingleInvocationFlow
- AccumulateFlow
- SlideFlow
- ASTGrouped
- Nby1Flow
**Active Provenance Capturing**

**Provenance Configuration**

- Specify Workflow
- Formulate Configuration types, clusters, sensors...
- Apply configuration
- Execute
- Monitor Validate

**Research Developers**
- Develop libs of ProvenanceTypes for Contextualisation and Precision

**Scientists (Workflow Users)**
- Combine and Assign ProvenanceTypes to WF Functions
- Enrich descriptions with semantic tags

**System Managers**
- Selective Lineage activation to narrow the focus of the lineage (metadata values-range)
- Tune the impact of provenance on the infrastructure (real-time systems)

**Domain Metadata**

**Contextualisation Types**
- SeismoType
- NetCDFType

**Precision of Data Derivations**

**Pattern Types**
- SingleInvocationFlow
- AccumulateFlow
- SlideFlow
- ASTGrouped
- Nby1Flow
Test Case: Seismic Rapid Assessment

Rapid Ground Motion Assessment (RA)

Choose/upload seismic wavespeed & mesh

Choose/upload seismic source (point or fault)

Gather observed data

run waveform simulation

Waveform Preprocessing

Ground Motion Parameters

Compare/integrate synthetic and observed ground motion data

Store data, metadata, provenance

Reusable Heterogeneous Tasks running at different scale. May require human monitoring and intervention
Test Case:
Seismic Rapid Assessment

- **Rapid Ground Motion Assessment (RA)**
  - Choose/upload seismic wavespeed & mesh
  - Choose/upload seismic source (point or fault)

  **MPI Simulation**
  - Run waveform simulation

  **Waveform Preprocessing**
  - Gather observed data

  **Ground Motion Parameters**

  **Data Analysis**
  - Compare/integrate synthetic and observed ground motion data

  **Data Staging**

Reusable Heterogeneous Tasks running at different scale. May require human monitoring and intervention.

Store data, metadata, provenance
Configuration Profile in JSON with Provenance Types

```json
{
    'provone:User': "aspinuso",
    's-prov:description': "provdemo",
    's-prov:workflowName': "waveform preprocessing pipeline",
    's-prov:workflowType': "seis:preprocessing",
    's-prov:WFExecutionInputs': [{'...}],
    's-prov:save-mode': 'service',
    's-prov:WFExecutionInputs': [{'...}],
    # defines the Provenance Types and Provenance Clusters for the Workflow's Components
    's-prov:componentsType':
        {'s-prov:componentsType':
            {'PE_ReadData':
                {'s-prov:type': ['SeismoType'],
                 's-prov:prov-cluster': 'seis:DataHandler'},
             'PE_taper':
                {'s-prov:type': ['SeismoType'],
                 's-prov:prov-cluster': 'seis:Processor'},
             'PE_remove_response':
                {'s-prov:type': ['SeismoType'],
                 's-prov:prov-cluster': 'seis:Processor'},
             'PE_plot_stream':
                {'s-prov:type': ['SeismoType'],
                 's-prov:prov-cluster': 'seis:Processor'},
             'StoreStream':
                {'s-prov:type': ['SeismoType'],
                 's-prov:prov-cluster': 'seis:DataHandler'}}
        }
}
```
Configuration Profile in JSON with Provenance Types

```json
{
    'prov:User': "aspinuso",
    'prov:system': "provdemo",
    'prov:workflowName': "waveform_preprocessing_pipeline",
    'prov:workflowType': "seis:preprocessing",
    'prov:WFExecutionId': "[...]",
    'prov:save-mode': 'service',
    'prov:WFExecutionInputs': "[...]",
    # defines the Provenance Types and Provenance Clusters for the Workflow's Components
    'prov:componentsType': {
        'prov:componentsType': {
            'PE_ReadData': {
                'prov:prov-type': ['SeismoType'],
                'prov:prov-cluster': ['seis:DataHandler'],
            },
            'PE_taper': {
                'prov:prov-type': ['SeismoType'],
                'prov:prov-cluster': ['seis:Processor'],
            },
            'PE_remove_response': {
                'prov:prov-type': ['SeismoType'],
                'prov:prov-cluster': ['seis:Processor'],
            },
            'PE_plot_stream': {
                'prov:prov-type': ['SeismoType'],
                'prov:prov-cluster': ['seis:Processor'],
            },
        },
    },
    'prov:delta': 0.01,
    'prov:calib': 1,
    'prov:sampling_rate': 100,
}
```
Waveform Preprocessing

functions encoded in Python
User Defined Metadata injection into Lineage traces

```
def plot_stream(stream, output_dir, tag):
    stats = stream[0].stats
    filename = "%s.%s.%s.png" %
                (stats['network'], stats['station'],
                 stats['channel'], tag)

    path = os.environ['STAGED_DATA'] + '/' + output_dir
    dest = os.path.join(path, filename)
    stream.plot(outfile=dest)

    prov = {'location': "file://" + socket.gethostname() +"/"+dest,
            'format': 'image/png',
            'metadata': {'origin': tag}}

    return {'_d4p_prov': prov, '_d4p_data': stream}
```
Monitor, search and analyse results through lineage

Runitme Monitoring
- Data produced
- Messages (Errors)
- Workers Nodes
- Event Times
- Runtime Changes

S-ProvFlow: https://gitlab.com/project-dare/s-ProvFlow
Ground Motion Parameters  seis:PGMCalculation

Pattern Type: Grouped Accumulator  (Stateful operator - ASTGrouped)
Combines intermediate inputs before producing results
- Develop & Register Workflows
- Scalable Workflow execution on containerised resources
- Lineage Capturing and visualisation

```
In [ ]:
# Local code
impl_id = F.create_peimpl_temp(desc="", code=In[2],
    parent_sig=pe_url, pckg="test_impl",
    name=" waveform_preprocessing",
    workspace=workspace_url,
    clone="", creds=creds)

print impl_id

In [ ]:
F.submit_d4p(impl_id=impl_id, pckg="test",
    workspace_id=workspace_id, pe_name=" waveform_preprocessing",
    token=F.auth(), creds=creds, n_nodes=6, no_processes=6, iterations=1)
```
Linking executions and semantic tagging
Exploring the Experiments’ space

Provenance of Multiple experiments with many stages

Visual analytics of data reuse between the workflows of the RA use case

Runs selected among those using the same station codes. (Contextual metadata)

**Vertices:** workflows execution *ids* colour-coded by user

**Edges:** data flows. Red and green edges for data input and output.

**Right half:** better descriptions yield the improved understanding, discovery and reuse of the results

[S-ProvFlow:](https://gitlab.com/project-dare/s-ProvFlow)
Provenance-aware Workspaces

Notebook API

A Web API to:

- **Create Notebook** Workspaces with the required libraries
- **Stage, pre-process** data onto active Workspaces (pre-built workflows)
  - Data staging history
  - Read-only and extensible raw data
- **Update** the Workspace libraries
- **On demand snapshots** of the Working Session (K8S volumes/Git/Binder)
- Data staging history
- Read-only and extensible raw data
- Update the Workspace libraries
- On demand snapshots of the Working Session (K8S volumes/Git/Binder)

- **Data Staging**
  - Raw Data Volume
  - Results Volume
  - Notebook pages
  - lib requirements
Provenance-aware Workspaces

Notebook API

A Web API to:

- **Create Notebook** Workspaces with the required libraries
- **Stage, pre-process** data onto active Workspaces (pre-built workflows)
  - Data staging history
  - Read-only and extensible raw data
- **Update** the Workspace libraries
- **On demand snapshots** of the Working Session (K8S volumes/Git/Binder)

Provenance-aware Workspaces

Luc Moreau et al. A Templating System to Generate Provenance
https://eprints.soton.ac.uk/405025/1/provtemplate.pdf

ProvenaceTemplate Catalogue
https://github.com/EnvriPlus-PROV/ProvTemplateCatalog
Conclusions & Future Work

- **Balanced automation and Active human contribution** in provenance capturing in Data-Intensive workflows

- **Provenance model** S-PROV, that accommodates complex lineage patterns

- **A conceptual design** based on reusable and combinable *Provenance Types* that lead to the *Provenance Configuration*

- **Services and tools** developed around our framework to control and evaluate the executions (DARE API, S-ProvFlow)

- **Coming Next!** Integration within Traceable Workspaces
Thanks!

http://project-dare.eu
Specify Workflow

Workflow encoded in Python

```python
def buildWorkflow():
    real_preprocess = create_processing_chain(proc['data_processing'])
    synt_preprocess = create_processing_chain(proc['synthetics_processing'])
    print(real_preprocess)
    graph = WorkflowGraph()
    read = ReadDataPE()
    read.name = 'data'
    read.output_units = proc['output_units']
    rotate_real = RotationPE('data')
    rotate_synt = RotationPE('synth')
    store_real = StoreStream('data')
    store_synt = StoreStream('synth')
    graph.connect(read, 'output_real', real_preprocess, 'input')
    graph.connect(read, 'output_synt', synt_preprocess, 'input')
    if proc['rotate_to_ZRT']:
        graph.connect(real_preprocess, 'output', rotate_real, 'input')
        graph.connect(synt_preprocess, 'output', rotate_synt, 'input')
        graph.connect(rotate_real, 'output', store_real, 'input')
        graph.connect(rotate_synt, 'output', store_synt, 'input')
    else:
        graph.connect(real_preprocess, 'output', store_real, 'input')
        graph.connect(synt_preprocess, 'output', store_synt, 'input')
    return graph

graph = buildWorkflow()

from dispel4py.visualisation import display
display(graph)
```
Why PROV Templates (as a service)

- **Templates foster discussions** on provenance relationships involving heterogeneous agents and resources (*Human vs System Concerns*).

- **Modelling of usable** and **re-usable** provenance scenarios (tailoring vs generalisation)

- **Remove the burden to hardcode** provenance editing
  (expansion tools/services)

Luc Moreau et al. A Templating System to Generate Provenance
[https://eprints.soton.ac.uk/405025/1/provtemplate.pdf](https://eprints.soton.ac.uk/405025/1/provtemplate.pdf)
ProvenaceTemplate Catalogue
[https://github.com/EnvriPlus-PROV/ProvTemplateCatalog](https://github.com/EnvriPlus-PROV/ProvTemplateCatalog)
[https://envriplus-provenance.test.fedcloud.eu/](https://envriplus-provenance.test.fedcloud.eu/)