dispel4py: An Open-Source Python library for Data-Intensive Seismology

Rosa Filgueira, Amrey Krause, Alessandro Spinuso,
Iraklis Klampanos, Peter Danecek, and Malcolm Atkinson

Roadmap

- Introduction
- Dispel4py features
- dispel4py basic concepts
- dispel4py advance concepts
- Verce project
- Seismic ambient noise cross-correlation
- Conclusions and future work

Introduction – What it is dispel4py ?

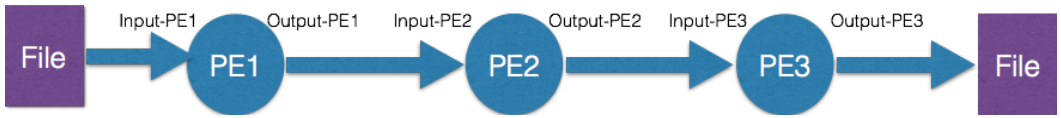
- New user-friendly tool
- Develop scientific methods and applications on local machines
- Run them at scale on a wide range of computing resources without making changes

dispel4py features

- **Stream-based**
  - Tasks are connected by streams and not by intermediate files
  - Multiple streams in & out
  - Optimization based on avoiding IO
- **Maps workflows dynamically** onto multiple enactment systems
- **Python** language for describing tasks and connections
- Reproducible and open scientific methods via a **registry and provenance mechanisms.**

dispel4py basic concepts – Processing element

- PEs represent the basic computational blocks of any dispel4Py workflow: algorithm, service, data transformation
- Shared: Storing them into the registry
- PEs can be seen as "Lego bricks" of tasks. Users can assemble them into workflow as they wish.
- General PE features
  - Consumes any number and types of input streams
  - Produce any number and types of output streams
  - It does some processing→ computational activity

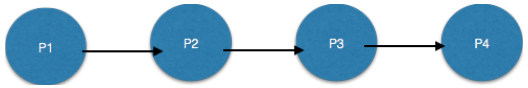


dispel4py basic concepts – Instance and graph

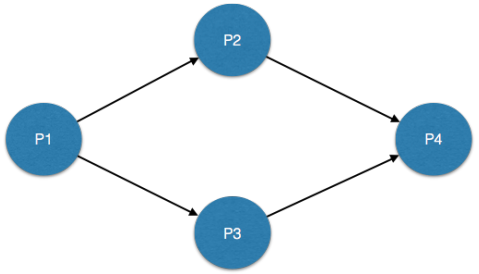
- Graph
  - Topology of the workflow: connections among PES
- Instance
  - Executable copy of a PE that runs in a process.
  - Each PE is translated into one or more instances in run-time

- Example of graphs

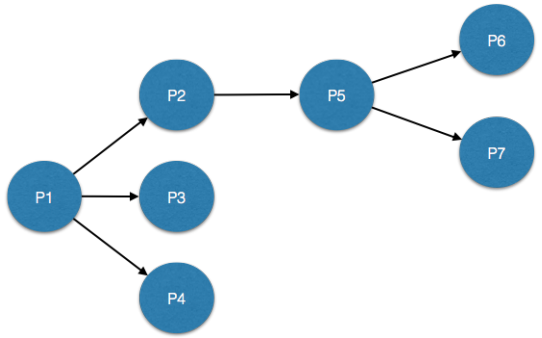
Pipeline



Split & Merge



Tree



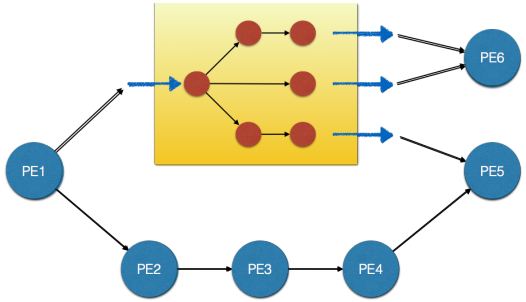
edinburgh data-intensive research

VERCE

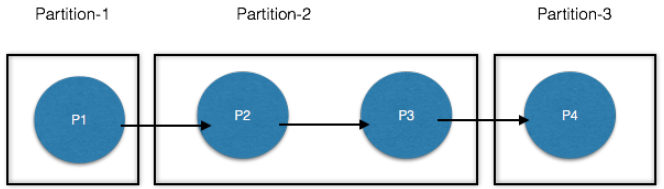
dispel4py basic concepts – Composite PE and partition

- Composite PE
  - Sub-workflow in a PE
  - Hides the complexity of an underlying process
  - Treated like any other PE
- Partition
  - PEs wrapped together
  - Run several PEs in a single process

- Example of Composite PE



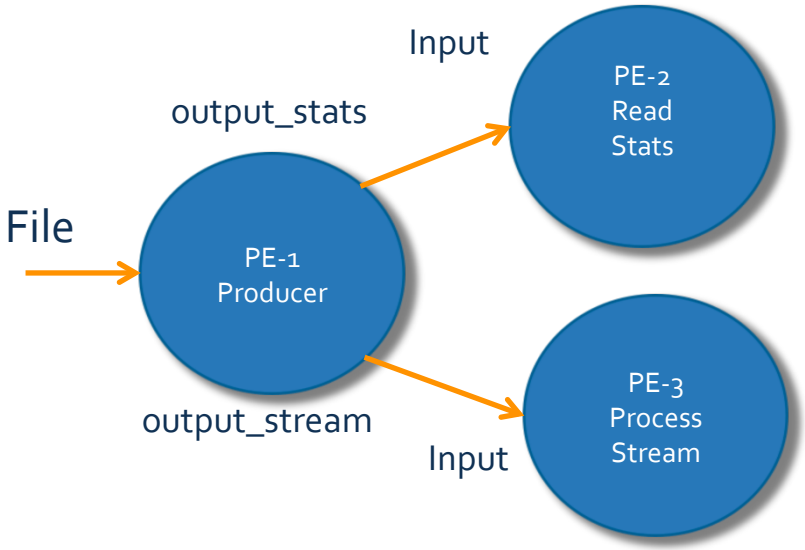
- Example of Partition



edinburgh data-intensive research

VERCE

dispelp4y basic concepts – Graph example



Users only have to implement:

- PEs
- Connections

```
from dispel4py.worflow _graph import WorkflowGraph
pe1 = Producer()
pe2 = ReadStats()
pe3 = ProcessStream()
graph= WorkflowGraph()
graph.connect (pe1, 'output_stats', pe2, 'input')
graph.connect (pe1, 'output_stream', pe3, 'input')
```

edinburgh data-intensive research

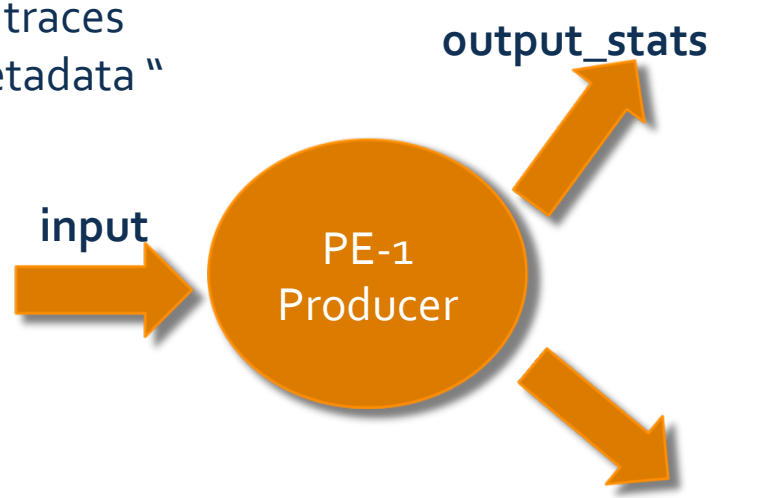
VERCE

dispel4py basic concepts -- Example of a PE

```
from dispel4py.core import GenericPE
from obspy import read

Class Producer(GenericPE):
   '''
   This PE reads a file that contains seismological traces
   and returns two outputs: obspy stream and metadata "
   def __init__(self):
      GenericPE.__init__(self)
      self._add_input('input')
      self._add_output('output_steam')
      self._add_output('output_stats')

    def process(self, inputs):
      data = inputs['input']
      st = read(data)
      return {'output_stream': st, 'output_stats': st[0].stats}
```



dispel4py – Advance concepts

- Mappings
  - Automatic and efficient parallelization to different parallel engines
  - Without any cost for users
- Provenance
  - Runtime analysis of the provenance information collected
  - Interactive validation of results
- Registry
  - Store workflows (and their PEs) with annotations

dispel4py advance concepts – Mappings

- **Sequential**
  - Sequential mapping for local testing
  - Ideal for local resources: Laptops and Desktops
- **Multiprocessing**
  - Python's multiprocessing library
  - Ideal for shared memory resources
- **MPI**
  - Distributed Memory, message-passing parallel programming model
  - Ideal for HPC clusters
- **STORM**
  - Distributed Real-Time computation System
  - Fault-tolerant and scalable

Run dispel4py workflows

Sequential mapping

>> dispel4py **simple** <name_dispy_graph>

Multi-process mapping

>> dispel4py **multi** <name_dispy_graph> **-n <number mpi_processes>**

MPI mapping

>> mpiexec -n <number mpi_processes> dispel4py **mpi** <name_dispy_graph>

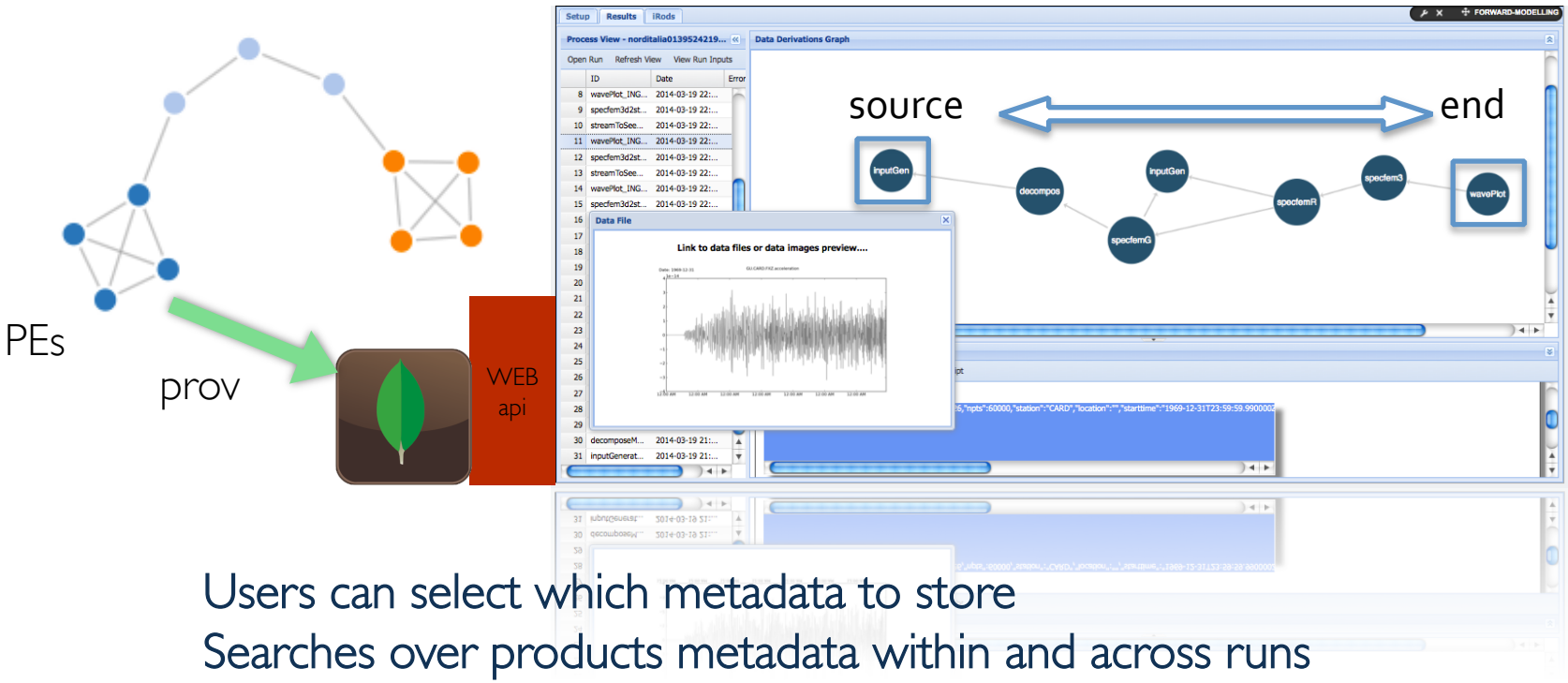
STORM mapping

>> python **storm_submission.py** <name_dispy_graph> <-m mode:
[remote, local, create]>

edinburgh data-intensive research

VERCE

dispel4py advance concepts – Provenance

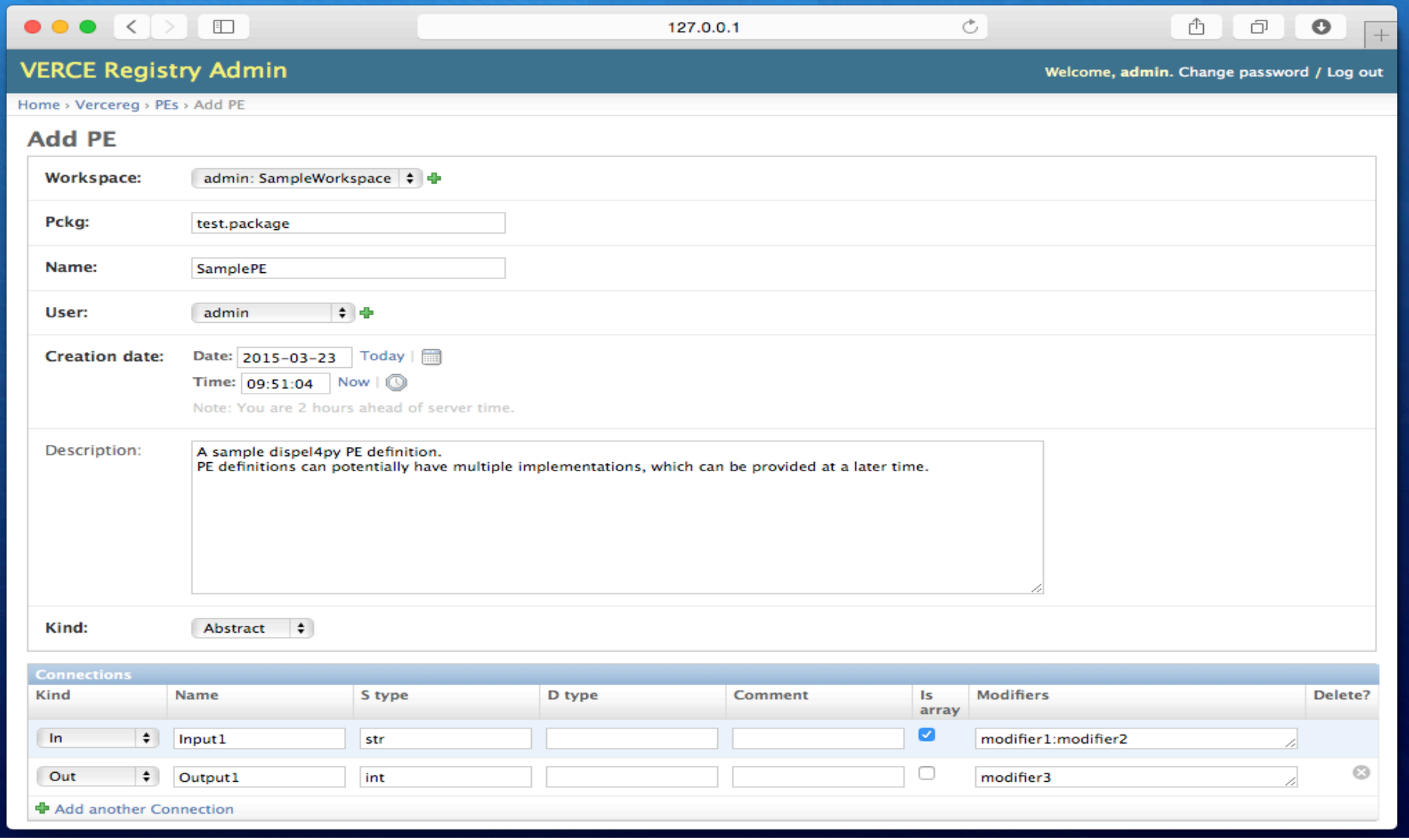


Users can select which metadata to store
Searches over products metadata within and across runs
Data download and preview
Capturing of Errors for Diagnostic purposes
Data Fabric: Multi directional navigations across data dependencies
W3C PROV-DM as reference model.

edinburgh data-intensive research

VERCE

dispel4py advance concepts – Registry



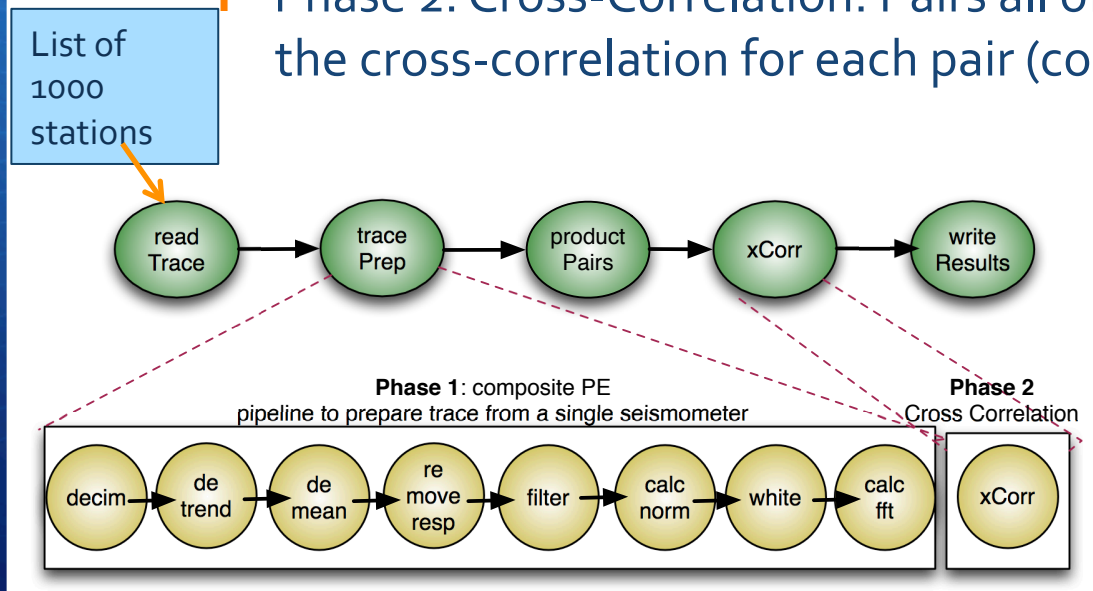
VERCE project

- The VERCE project provide a framework to the seismological community to exploit the increasingly large volume of seismological data :
  - Support data-intensive and HPC applications
  - e-Science Gateway for submitting applications
  - Distributed and diversified data sources
  - Distributed HPC resources on Grid, Cloud and HPC clusters

- Use cases – dispel4py :
  - **Seismic Noise Cross-Correlation**
  - MISFIT

Seismic ambient noise cross-correlation – 1000 stations

- Data intensive problem and it is commonly used in seismology
  - Phase 1- Preprocess: Time series data (traces) from seismic stations are preprocessed in parallel
  - Phase 2: Cross-Correlation: Pairs all of the stations and calculates the cross-correlation for each pair (complexity O(n2)).



Input data:
1000 stations as input data (150 MB)
Output data:
499,500 cross-correlations (39GB)

Evaluations – Computing resources

Computing Resources	Terracorrelator	SuperMuc	Amazon EC2	EDIM1
Type	**Shared-memory**	**Cluster**	**Cloud**	**Cloud**
Enactment Systems	MPI, multi	MPI, multi	MPI, Storm, multi	MPI, Storm, multi
Nodes	1	16	18	14
Cores per Node	32	16	2	4
Total Cores	**32**	**256**	**36**	**14**
Memory	2TB	32GB	4GB	3GB

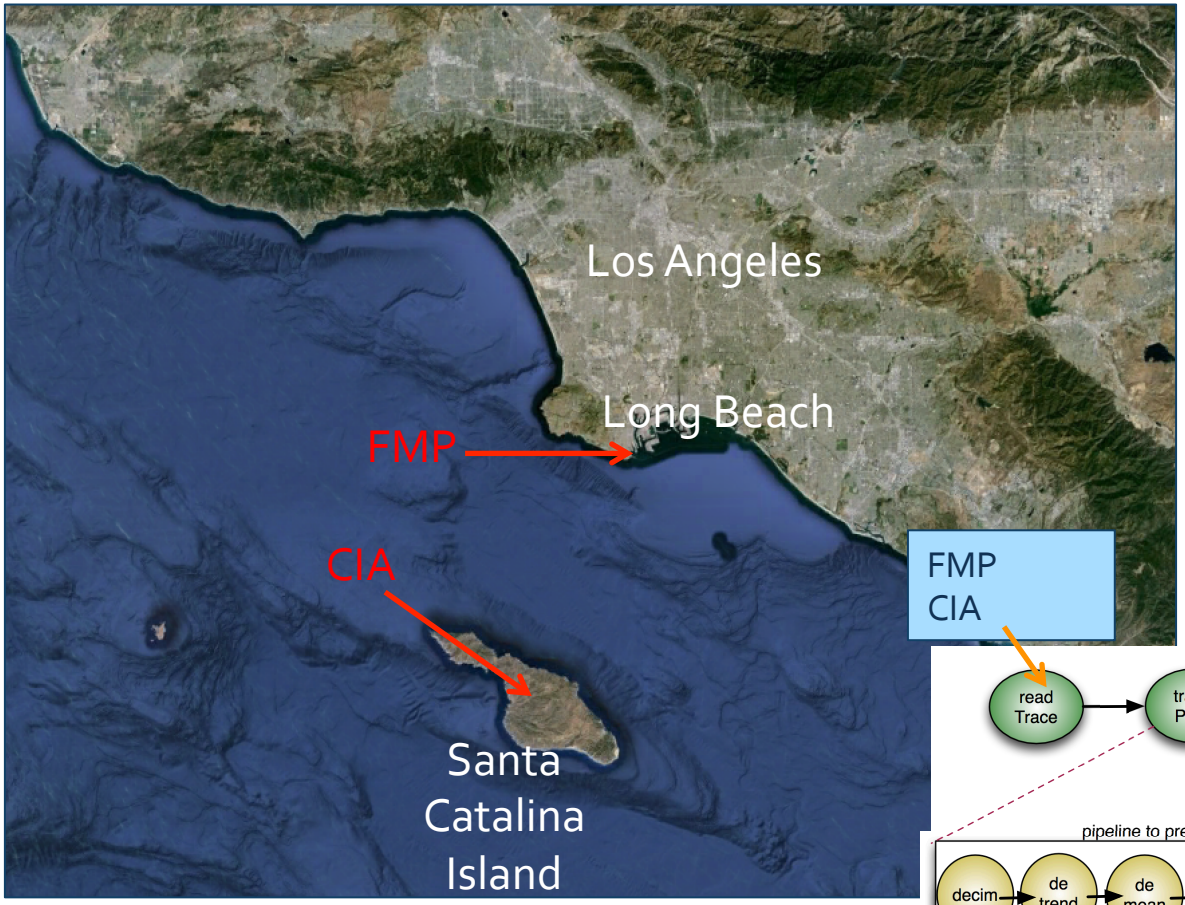
Evaluations – Measures in seconds

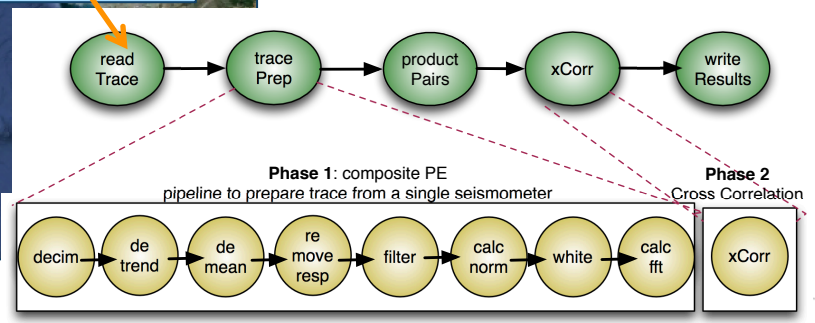
Computing Resources / Enactment System	Terracorrelator (32 cores)	SuperMuc (256 cores)	Amazon (36 cores)	EDIM1 (14 cores)
MPI	2510.22 (~42minutes)	1093.16 (~19minutes)	16862.73 (~5hours)	38656.94 (~11 hours)
multi	2665.10 (~44minutes)		27898.89 (~8 hours)	
Storm				120077.123 (~33 hours)

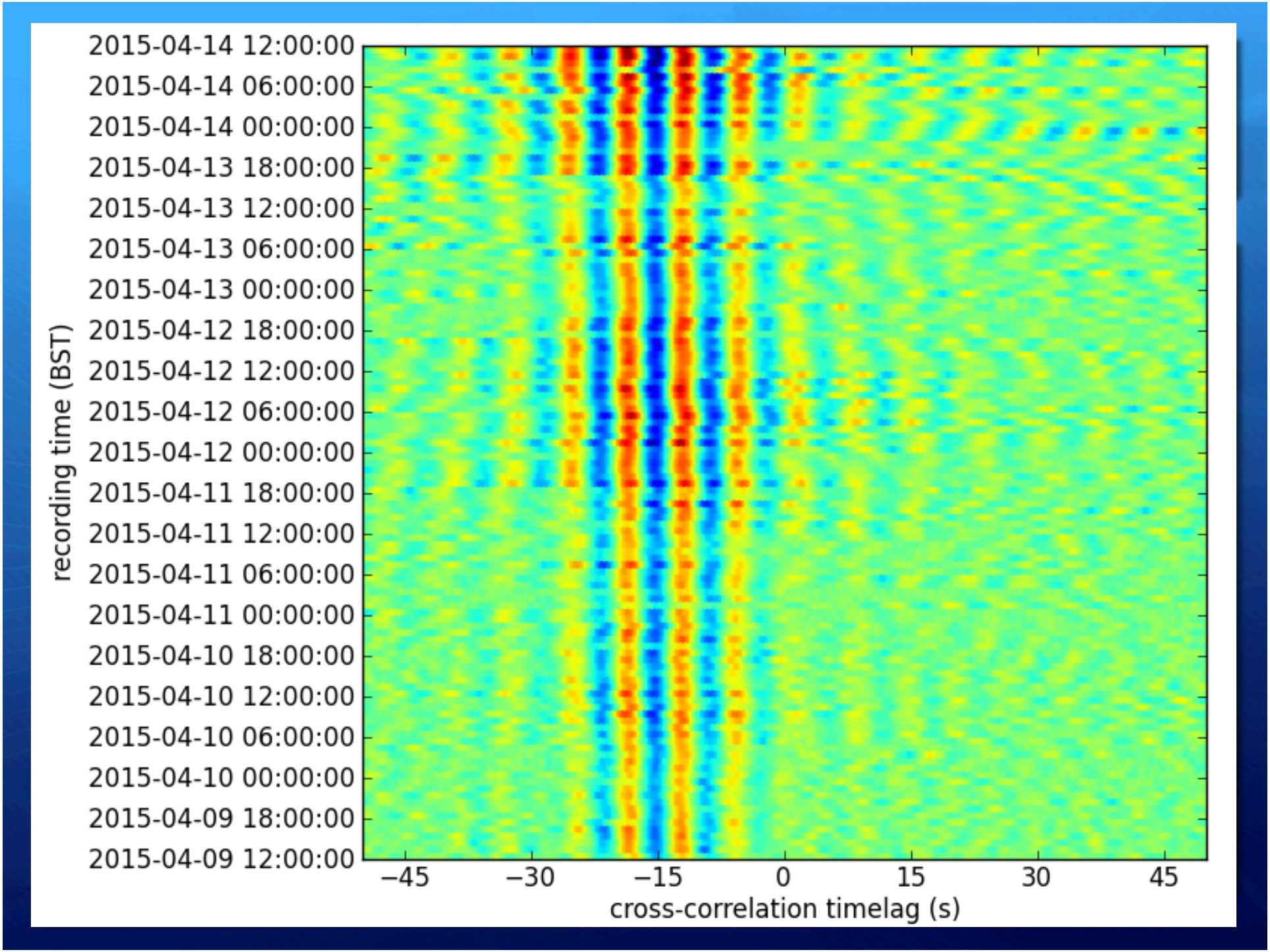
edinburgh data-intensive research

VERCE

Near real-time continuous correlations of two stations: CI-FMP and CI-CIA from the Caltech Regional Seismic Network






2015-04-14 12:00:00
2015-04-14 06:00:00
2015-04-14 00:00:00
2015-04-13 18:00:00
2015-04-13 12:00:00
2015-04-13 06:00:00
2015-04-13 00:00:00
2015-04-12 18:00:00
2015-04-12 12:00:00
2015-04-12 06:00:00
2015-04-12 00:00:00
2015-04-11 18:00:00
2015-04-11 12:00:00
2015-04-11 06:00:00
2015-04-11 00:00:00
2015-04-10 18:00:00
2015-04-10 12:00:00
2015-04-10 06:00:00
2015-04-10 00:00:00
2015-04-09 18:00:00
2015-04-09 12:00:00
recording time (BST)
−45
−30
−15
0
15
30
45
cross-correlation timelag (s)

Conclusions and future Work

- Novel Python library for **streaming and data-intensive processing**
- **Novelty**
  - Users express their computational activities
  - Same workflow can be automatically executed in several parallel systems
  - dispel4py is easy to use, allows users to share , re-use Pes and collect provenance information.
- **Future**
  - Support for PE failures
  - New mapping: Apache Spark
  - diagnostic tools to select the best computing resource and mapping

Thanks & Questions

- Documentation
  - http://dispel4py.org
- Source code
  - https://github.com/dispel4py/dispel4py
- Installation
  - **pip install dispel4py**
- Contact emails
  - Rosa Filgueira: rosa.filgueira@ed.ac.uk
  - Amy Krause: akrause@staffmail.ed.ac.uk
  - Alessandro Spinuso: spinuso@knmi.nl