Virtual Earthquake and seismology Research Community e-science environment in Europe
Project 283543 – FP7-INFRASTRUCTURES-2011-2 – www.verce.eu – info@verce.eu

dispel4py: An Agile Framework for Data-Intensive eScience

11Th IEEE eScience 2015
2nd September 2015

Rosa Filgueira e
University of Edin

Roadmap

Introduction

dispel4py features

dispel4py basic concepts

dispel4py advance concepts

dispel4py workflows

Evaluations

Current work

Conclusions and future work

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

What it is dispel4py ?

pen source project: www.dispel4py.org & https://github.com/dispel4py/dispel4py & pip install dispel4py

blications:

- DISCS-2014 workshop, SC14, 2014
- IJHPCA journal, “Data-Intensive High Performance Computing” Special Issue, 2015
- 11th IEEE eScience Conference, 2015
- Book Chapter in “Conquering Big Data Using High Performance”, 2015

any users:

- Computational Seismologists, VERCE project.
- Astrophysicists

ainings:

- General audience, OSDC PIRE, 2015, UvA
- VERCE trainings: Munich and Liverpool, 2015

any contributors:

- University of Edinburgh: EPCC, DIR Group, Master Students
- KNMI

dispel4py features

Stream-based

- Tasks are connected by streams. Not by intermediate files
- Optimization based on avoiding IO.
- Multiple streams in & out

Data-flow oriented

- Data is transformed among the tasks

Maps workflows dynamically onto multiple enactment systems

Python language for describing tasks and connections

Encourage the reproducibility & re-usability of workflows:

Processing element

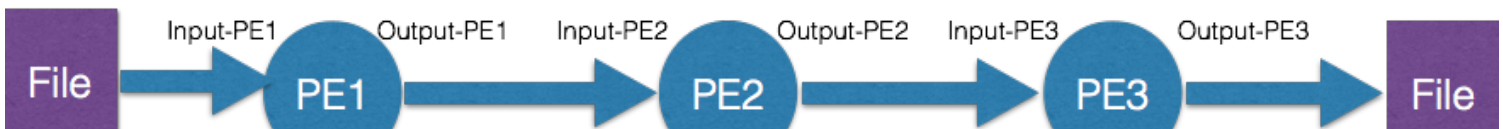
- PEs represent the basic computational:

  Algorithm, service, data transformation
- Shared: Storing them into the registry
- PEs ~ “Lego bricks” of tasks.

  Users can assemble them into workflow as they wish.

- Consume/Produce any number

  and types of input/output streams



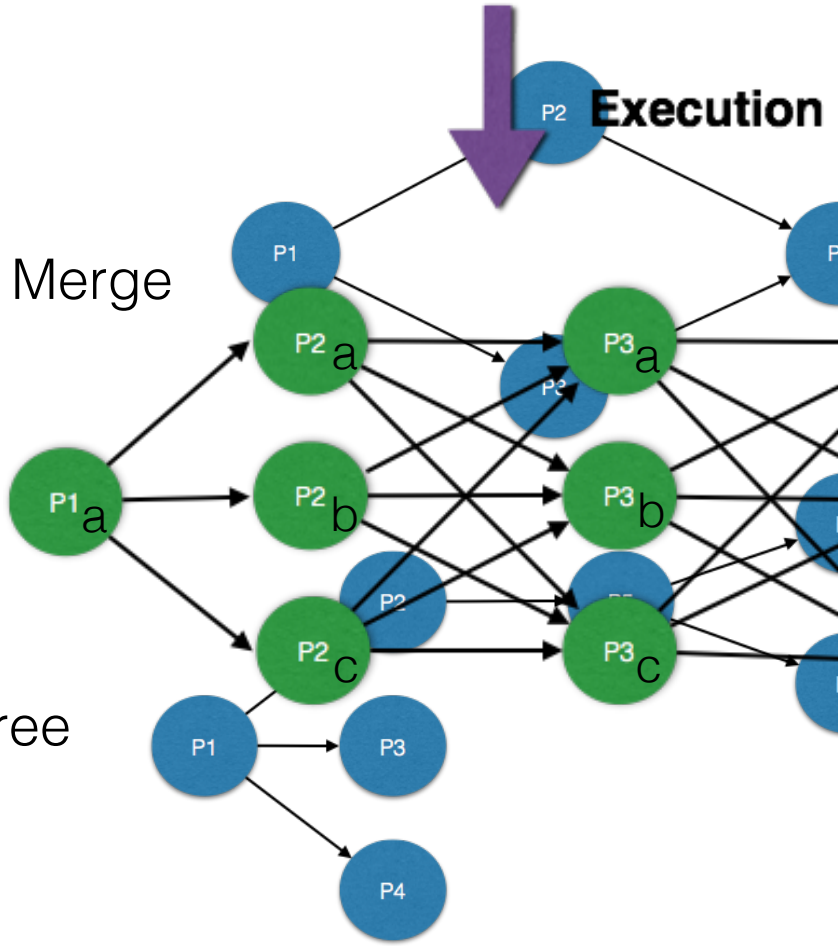
nstance and graph

- Graph
  - **Topology** of the workflow: connections among PES
  - It is what **user need** to "**think**" about.
  - Abstract workflow
- Instance
  - **Executable copy** of a PE that runs in a process.
  - Each PE is translated into one or more instances in run-time
  - dispel4py **does it** for you
  - Concrete workflow

+ Example of graphs

Pipeline

Split & Merge

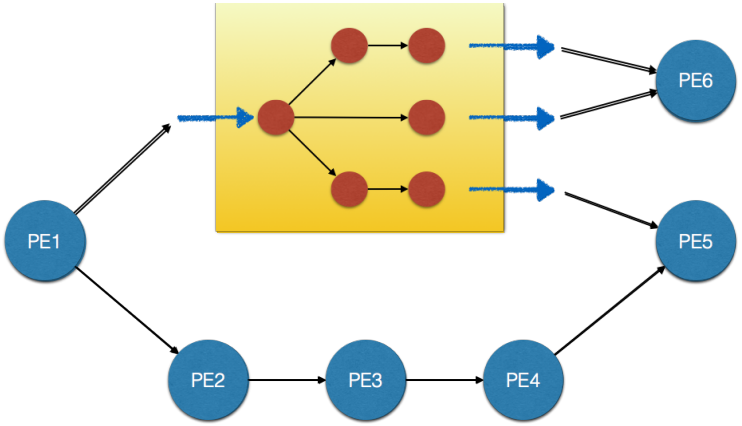


Tree

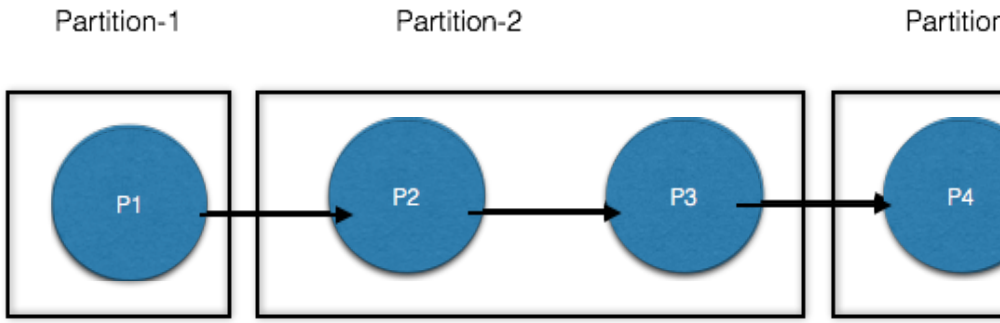
Composite PE and partition

- Composite PE
  - Sub-workflow in a PE
  - Hides the complexity
  - Treated like any other PE
- Partition
  - PEs wrapped together
  - Run several PEs in a 1 process

+ Example of Composite PE



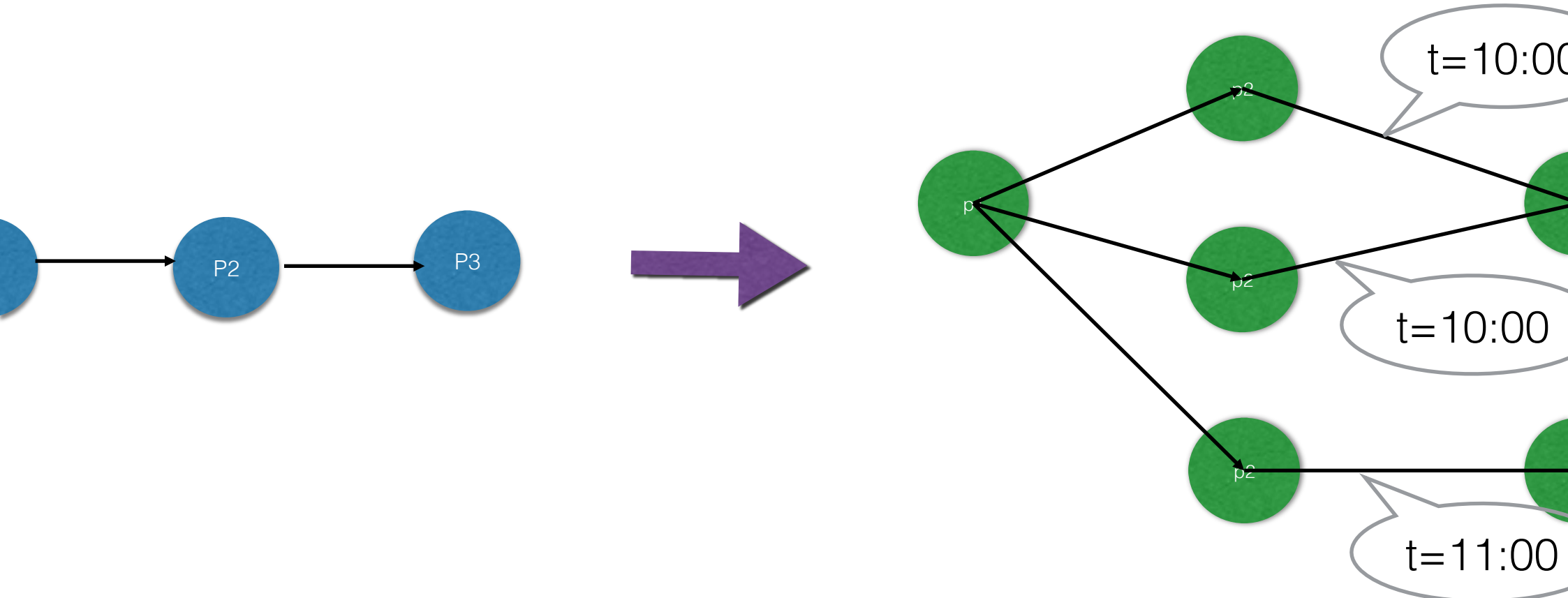
+ Example of Partition



Groupings

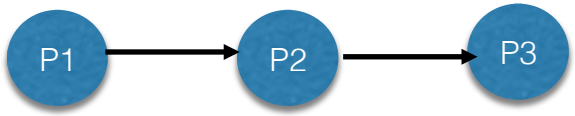
Grouping by" a feature (MapReduce)

data items that satisfy the same feature are guaranteed to be livered to the same **instance** of a PE

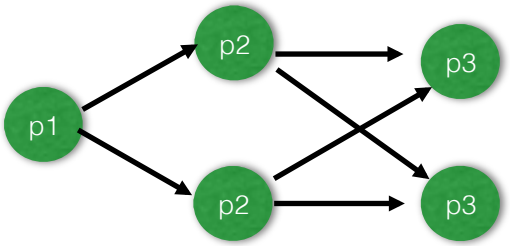


Groupings

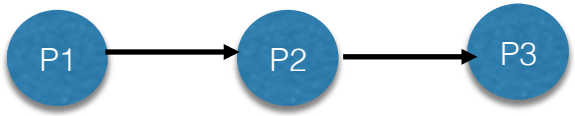
-To-All



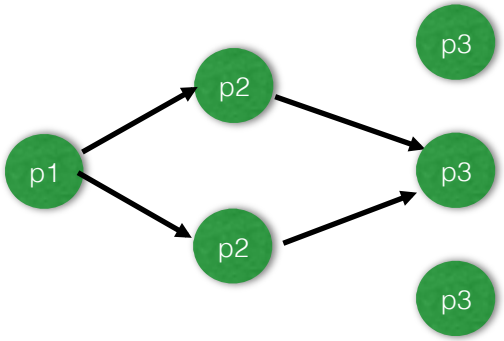
P3 - grouping “all”:
Instances send copies of their output data to **all** the connected instances



bal



P3 - grouping “global”:
All the instances send all the data to **one** connected instance



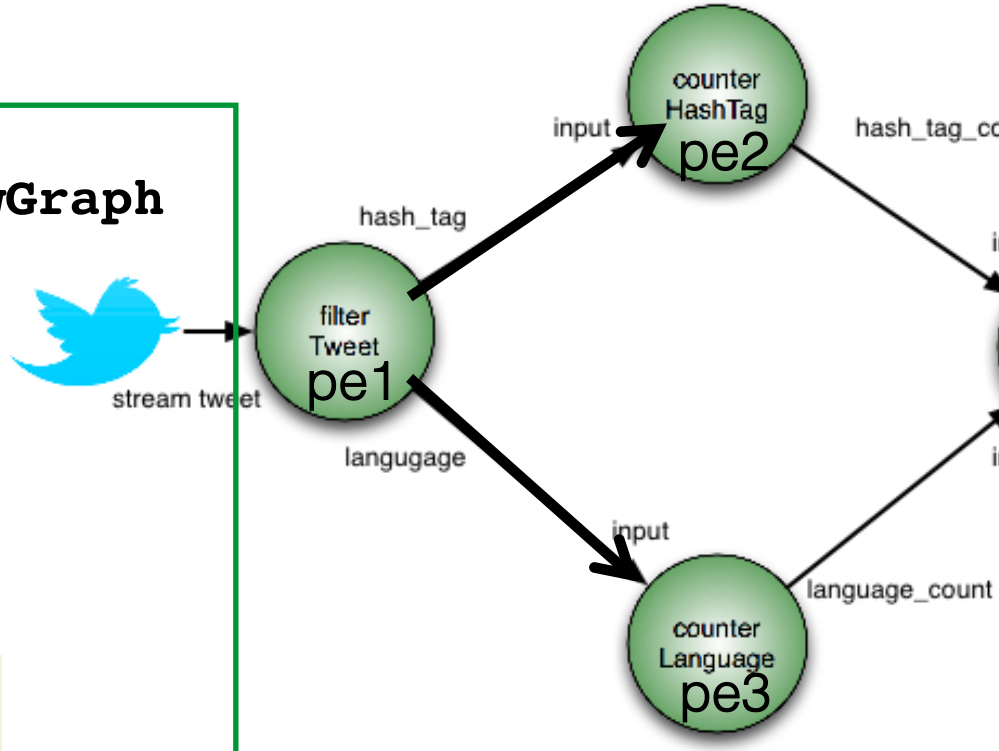
Example of a dispel4py workflow

```
ispel4py.workflow_graph import WorkflowGraph


filterTweet ()
counterHashTag ()
counterLanguage ()
statistics ()

= WorkflowGraph ( )

connect(pe1,'hash_tag',pe2,'input')
connect(pe1,'language',pe3,'input')
connect(pe2,'hash_tag_count',pe4,'input1')
connect(pe3,'language_count',pe4,'input2')
```

PEs objects

Graph

Connections

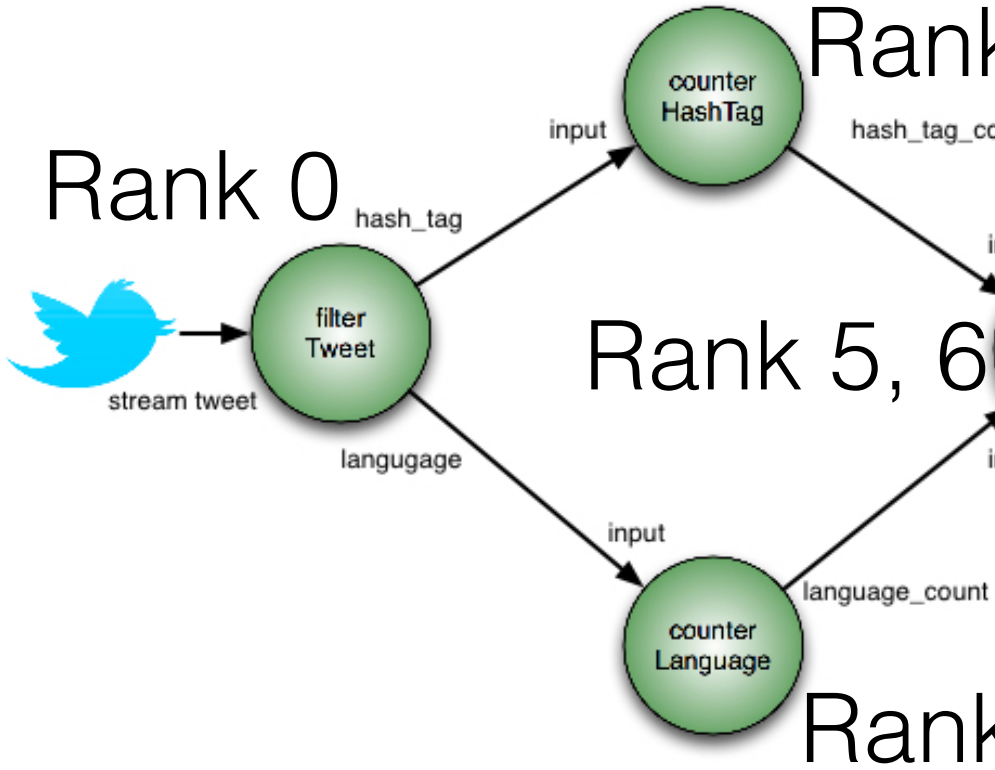


Users only have to implement:

- PEs
- Connections

Example of a "concrete" dispel4py workflow

```
1):
nk == 0:
ad (data)
sh_tag= Filter(data, hash_tag)
nguage = Filter(data, language)
I_Isend(hash_tag, rank → 1,2)
I_Isend(language, rank → 3,4)

nk ==1 or rank == 2 :
I_Recv(input, 0->rank)
sh_tag_count = Count(input)
I_Isend(hash_tag_count, rank → 5,6)

nk == 3 or rank == 4:

nk == 5 or rank == 6:
I_Recv(input1, 1,2 → rank)
```



Users only have to implement:

- PEs
- Connections

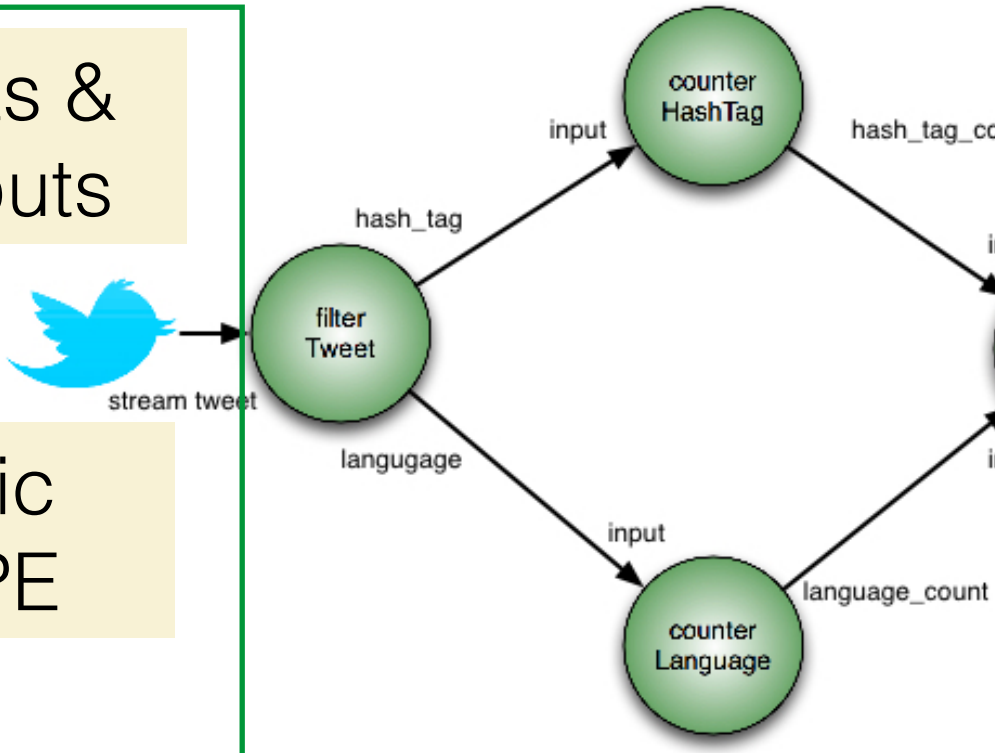
Example of a PE

```
filterTweet(GenericPE):
init__( self ):
nericPE.init (self)
lf.add_output('hash_tags ')
lf.add_output('language')

ocess ( self , inputs ):
itterDataFile= inputs['input ']
eet file = open(twitterDataFile)
r line in tweet file :
  tweet = json.loads( line )
  language = ' '
  hashtags =[]
  language = tweet[u'lang '].encode('utf -8')
  text = tweet[u'text '].encode('utf-8')
  hashtags=re.findall(r"#(\w+)", text)

  self.write('hash tag', hashtags)
  self.write('language', language)
```

Inputs & outputs

Logic of PE

Stream out



Users only have to implement:

- PEs
- Connections

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

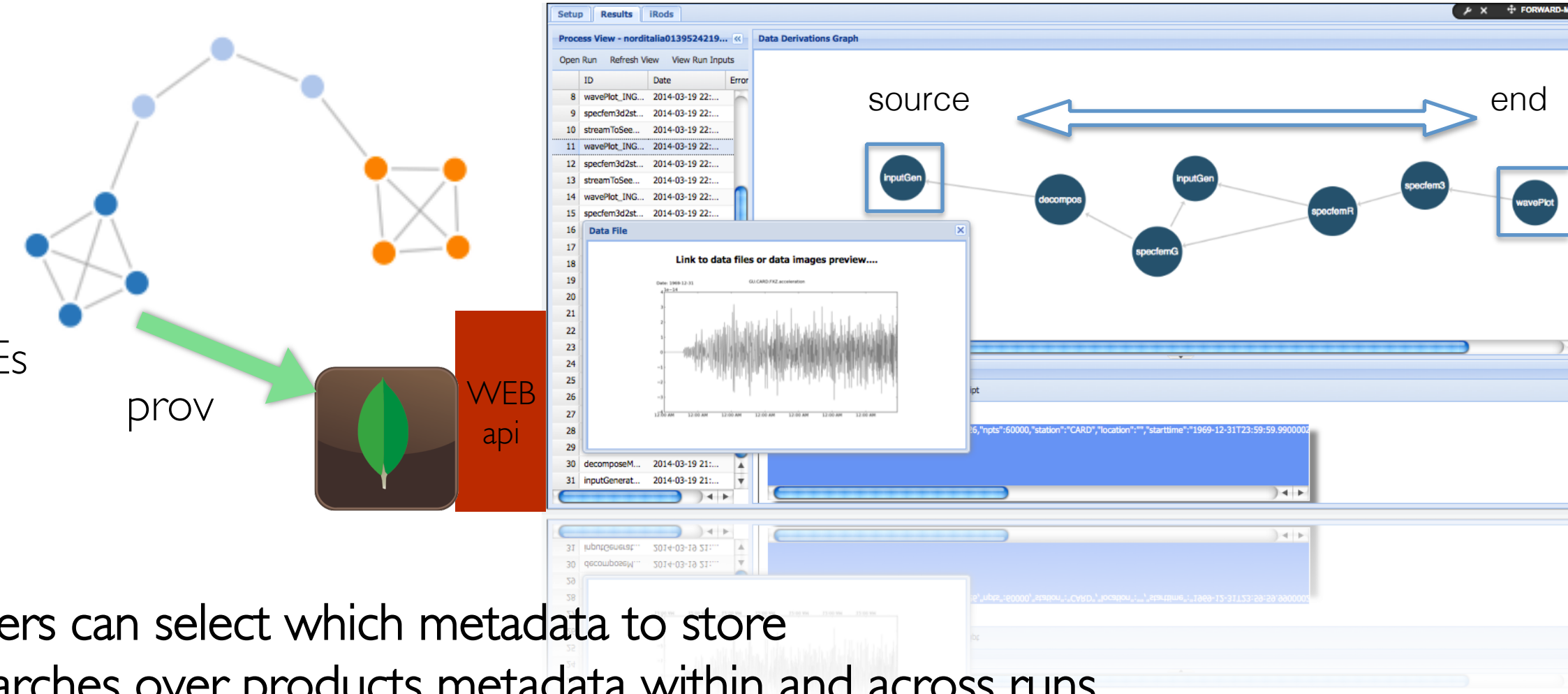
- Distributed Memory, message-passing parallel programming model
- Ideal for HPC clusters

STORM

- Distributed Real-Time computation System
- Fault-tolerant and scalable
- Runs all the time

Spark (Prototype)

Provenance



ers can select which metadata to store

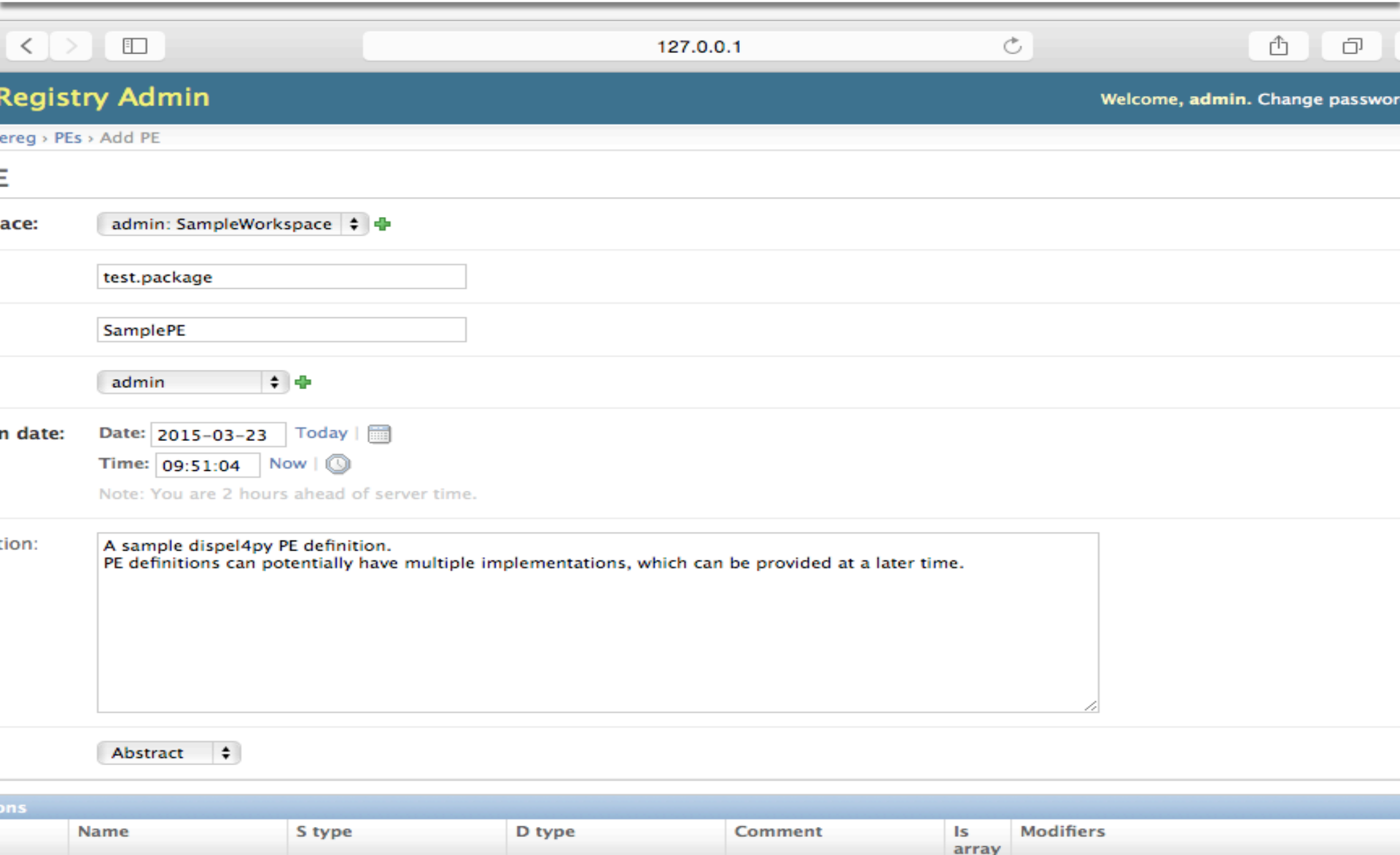
arches over products metadata within and across runs

ta download and preview

pturing of Errors for Diagnostic purposes

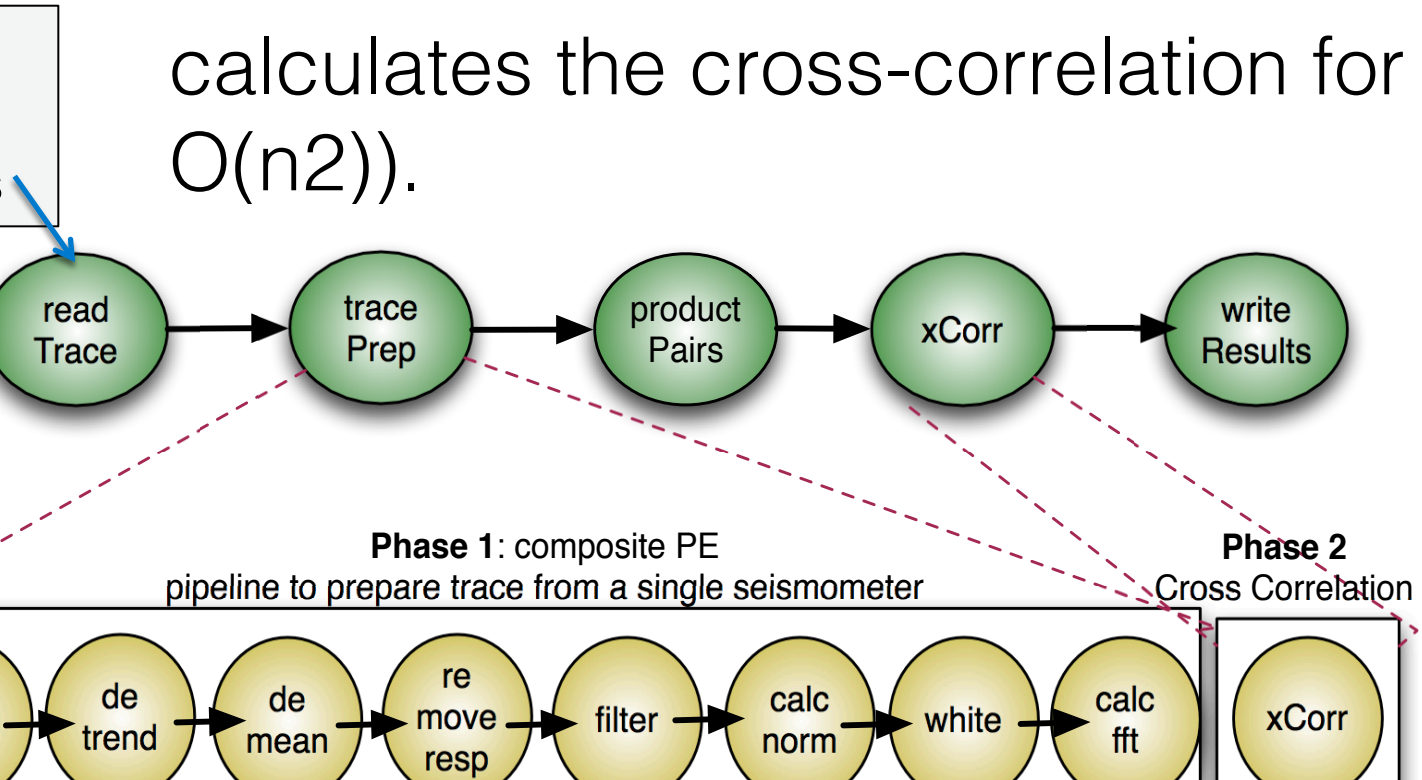
ta Fabric: Multi directional navigations across data dependencies

Registry


127.0.0.1
Registry Admin
Welcome, admin. Change passwor
ereg › PEs › Add PE
E
ace:
admin: SampleWorkspace
test.package
SamplePE
admin
n date:
Date: 2015-03-23
Today
Time: 09:51:04
Now
Note: You are 2 hours ahead of server time.
tion:
A sample dispel4py PE definition.
PE definitions can potentially have multiple implementations, which can be provided at a later time.
Abstract
ons
Name
S type
D type
Comment
Is array
Modifiers

Seismology, Cross Correlation

- Data intensive problem and it is commonly used in seismolog
  - Phase 1- Preprocess: Time series data (traces) from seism 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)).

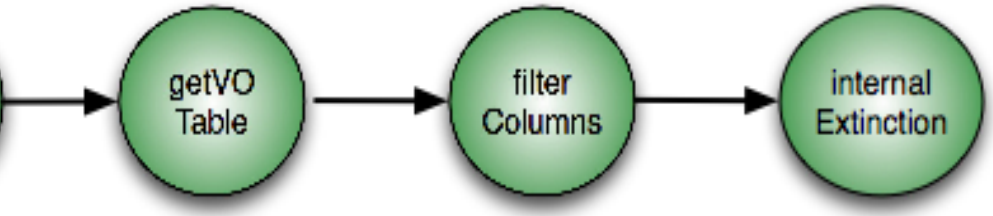


Input data:
1000 stations as input data (1
Output data:
499,500 cross-correlations (3

xcorr

Astrophysics, Internal Extintion

culates the Internal Extinction of the Galaxies from the AMIGA catalo



_ext

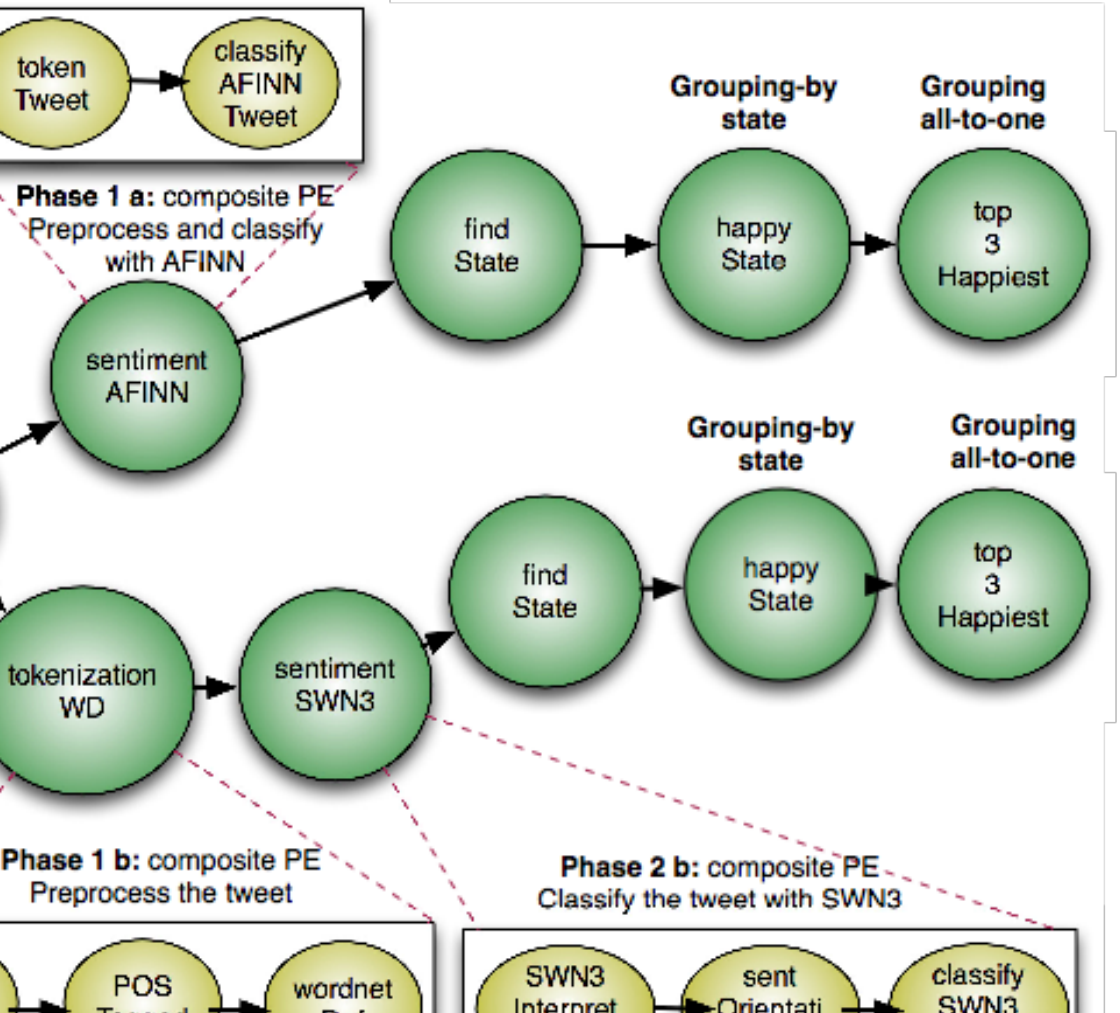
- The first PE **reads an input file (1051 galaxies)**:
  - **Right Ascension** (Ra)
  - **Declination** (Dec)
- The second PE **queries** a **VO service** for each galaxy
- The results are filtered by filterColumns
- Their internal extinction is calculated b internalExtinction PE.

Social Computing, Sentiment Analysis

basic sentiment analyses (126,826 tweets (500MB)) by applying two lex

AFINN

SentiWordNet



- The findState PE searches the US state
- The HappyState PE aggregates the sen scores of tweets from the same state
- The last PE determines which are the to three happiest states.

sentiment

Computing resources

mputing sources	Terracorrelator	SuperMuc	Amazon EC2	EDIM1
Type	Shared-memory	Cluster	Cloud	Cloud
actment ystems	MPI, multi	MPI, multi	MPI, Storm, multi	MPI, Storm multi
Nodes	1	16	18	14
res per Node	32	16	2	4
al Cores	32	256	36	14
emory	2TB	32GB	4GB	3GB
rkflows	xcorr, int_ext,	xcorr,	xcorr	xcorr, int_e

Performance measures

	Terracorrelator (32 cores/Node)	SuperMuc (256 cores 16 cores/node)	Amazon (36 cores 2 cores/node)	EDIM1 (14 cores 4 cores/node)
	1501.32 (~25minutes)	1093.16 (~19minutes)	16862.73 (~5hours)	38656.94 (~11 hours)
	1332 .20 (~23minutes)			
			27898.89 (~8 hours)	120077.123 (~33 hours)

xcorr:

1000 stations
Input 150MB
Output 39GB

Terracorrelator (32 cores/node)	EDIM1 (14 cores, 4 cores/node)
31.60	96.12
14.50	101.2
	30.2

Mode	Terracorrelator (32 cores/node)	EDIM1 (14 cores)	SuperMuc (256 cores 16 cores/no
MPI	925.04 (~15minutes)	32396.04 (~8hours)	2057.59 (~34minu
multi	971.04 (~16minutes)		3954.63 (~1hour

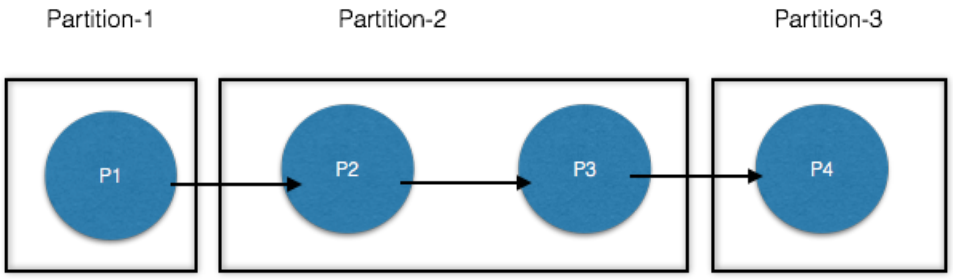
sentiment:

Current work

Run-time Stream Adaptive Compression

Diagnosis tool

- Quality check and Failure detector
- How to partition the workflow automatically
- How many processes execute each partition



Conclusions and Future work

Python library for streaming and data-intensive processing

- Users express their computational activities
- Same workflow executed in several parallel systems
- Easy to use, open, and encourage sharing the methods & applications

Future work

- Support for PE failures
- Select the best computing resource and mapping

nstallations and Links

- This is all you need:

pip install dispel4py

- Web site http://dispel4py.org/
- GitHub: https://github.com/dispel4py/dispel4py
- Documentation: http://dispel4py.org/documentation/

hanks and Questions

Contact emails

- Rosa Filgueira: rosa.filgueira@ed.ac.uk
- Amy Krause: a.krause@epcc.ed.ac.uk
- Alessandro Spinuso: spinuso@knmi.nl
- Malcolm Atkinson: Malcolm.Atkinson@ed.ac.uk