dispel4py: An Agile Framework for Data-Intensive eScience

11th IEEE eScience 2015
2nd September 2015

Rosa Filgueira et al.
University of Edinburgh
Roadmap

- Introduction
- dispel4py features
- dispel4py basic concepts
- dispel4py advance concepts
- dispel4py workflows
- Evaluations
- Current work
- 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
Introduction –
What it is dispel4py?

- Open source project: [www.dispel4py.org](http://www.dispel4py.org) & [https://github.com/dispel4py/dispel4py](https://github.com/dispel4py/dispel4py)

- Publications:
  - DISCS-2014 workshop, SC14, 2014
  - 11th IEEE eScience Conference, 2015
  - Book Chapter in “Conquering Big Data Using High Performance”, 2015

- Many users:
  - Computational Seismologists
  - Astrophysicists
  - BioInformatics

- Many contributors:
  - University of Edinburgh: EPCC, DIR Group, Master Students
  - KNMI
  - LMU
  - University of Malaysia
dispel4py features

- **Stream-based**
  - Tasks are connected by streams. 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

- Modular & Reproducible & Open scientific methods via a **registry and provenance tools**.
dispel4py basic concepts – 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.
- 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
  - It is what user have to “think” about.
- **Instance**
  - Executable copy of a PE that runs in a process.
  - Each PE is translated into one or more instances in run-time

4 PEs & 10 processes

+ Example of graphs

- Pipeline
- Split & Merge
- Tree
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
“Grouping by” a feature (MapReduce)
All data items that satisfy the same feature are guaranteed to be delivered to the same instance of a PE.
dispel4py basic concepts – Groupings

One-To-All

P3 - grouping “all”:
P2 instances send copies of their output data to all the connected instances

Global

P3 - grouping “global”:
All the instances of P2 send all the data to one instance of P3
dispelp4y basic concepts – Example of a dispel4py workflow

from dispel4py.workflow_graph import WorkflowGraph

pe1 = filterTweet ()
pe2 = counterHashTag ()
pe3 = counterLanguage ()
pe4 = statistics ()

graph = WorkflowGraph ()

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

Users only have to implement:
• PEs
• Connections
dispelp4y basic concepts– Example of a PE

Class filterTweet(GenericPE):
def __init__(self):
    GenericPE.init(self)
    self.add_output('hash_tags ')
    self.add_output('language')

def process(self, inputs):
    twitterDataFile = inputs['input ']
    tweet_file = open(twitterDataFile)
    for 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)

Users only have to implement:
• PEs
• Connections

Inputs & outputs
Logic of PE
Stream out data
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
  • Runs all the time

• **SPARK (Prototype)**
  • HDFS, Layer on top of Hadoop
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.
The VERCE project provides 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
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 $O(n^2)$).

List of 1000 stations

**Input data:**
1000 stations as input data (150MB)

**Output data:**
499,500 cross-correlations (39GB)

**dispel4py workflows - Seismology, Cross Correlation**

VERCE delivers a productive e-Science environment for seismology research, Tuesday, 1st September 2015.
Calculates the Internal Extinction of the Galaxies from the AMIGA catalogue

- 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 PE.
- Their internal extinction is calculated by the internalExtinction PE.
Two basic sentiment analyses (126,826 tweets (500MB)) by applying two lexicons:

- **AFINN**
- **SentiWordNet**

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

https://github.com/linkTDP/BigDataAnalysis_TweetSentiment
# Evaluations – Computing resources

<table>
<thead>
<tr>
<th>Computing Resources</th>
<th>Terracorrelator</th>
<th>SuperMuc</th>
<th>Amazon EC2</th>
<th>EDIM1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Shared-memory</td>
<td>Cluster</td>
<td>Cloud</td>
<td>Cloud</td>
</tr>
<tr>
<td>Enactment Systems</td>
<td>MPI, multi</td>
<td>MPI, multi</td>
<td>MPI, Storm, multi</td>
<td>MPI, Storm, multi</td>
</tr>
<tr>
<td>Nodes</td>
<td>1</td>
<td>16</td>
<td>18</td>
<td>14</td>
</tr>
<tr>
<td>Cores per Node</td>
<td>32</td>
<td>16</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Total Cores</td>
<td>32</td>
<td>256</td>
<td>36</td>
<td>14</td>
</tr>
<tr>
<td>Memory</td>
<td>2TB</td>
<td>32GB</td>
<td>4GB</td>
<td>3GB</td>
</tr>
<tr>
<td>Workflows</td>
<td>xcorr, int_ext, sentiment</td>
<td>xcorr, sentiment</td>
<td>xcorr</td>
<td>xcorr, int_ext, sentiment</td>
</tr>
</tbody>
</table>
### Evaluations – Performance measures

<table>
<thead>
<tr>
<th>Mode</th>
<th>Terracorrelator (32 cores)</th>
<th>SuperMuc (256 cores)</th>
<th>Amazon (36 cores)</th>
<th>EDIM1 (14 cores 4 shared)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI</td>
<td>1501.32 (~25minutes)</td>
<td>1093.16 (~19minutes)</td>
<td>16862.73 (~5hours)</td>
<td>38656.94 (~11 hours)</td>
</tr>
<tr>
<td>multi</td>
<td>1332 .20 (~23minutes)</td>
<td></td>
<td>27898.89 (~8 hours)</td>
<td></td>
</tr>
<tr>
<td>Storm</td>
<td></td>
<td></td>
<td></td>
<td>120077.123 (~33 hours)</td>
</tr>
</tbody>
</table>

**xcorr:**
1000 stations
Input 150MB
Output 39GB

<table>
<thead>
<tr>
<th>Mode</th>
<th>Terracorrelator (32 cores)</th>
<th>EDIM1 (14 cores, 4 shared)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI</td>
<td>31.60 (~15minutes)</td>
<td>96.12 (16minutes)</td>
</tr>
<tr>
<td>multi</td>
<td>14.50 (~16minutes)</td>
<td>101.2 (~16minutes)</td>
</tr>
<tr>
<td>Storm</td>
<td></td>
<td>30.2 (~16minutes)</td>
</tr>
</tbody>
</table>

**int_ext:**
1050 galaxies

**sentiment:**
126,82 tweets
Input 500MB
Current work

- Run-time Stream Adaptive Compression
- Diagnosis tool
  - 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 Pes

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

- This is all you need:
  
  `pip install dispel4py`

- GitHub: [https://github.com/dispel4py/dispel4py](https://github.com/dispel4py/dispel4py)
- Documentation: [http://dispel4py.org/documentation/](http://dispel4py.org/documentation/)
Thanks 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