Delivering easy-to-use frameworks to empower data-driven applications on HPC environments

Asterism: Pegasus and dispel4py for data-intensive science

Rosa Filgueira, BGS-NERC, UK
Rafael Ferreira da Silva, ISI-USC, US
Ewa Deelman, ISI-USC, US
Amrey Krause, EPCC-UoE, UK
Malcolm Atkinson, Informatics-UoE, UK
Introduction

Big Data Sciences Era, Data Intensive Computing applications

*Scientific communities*

- Fetch large set of input data
- Apply *methods* between live and archive data
- Move data between stages
- Perform *computations* for simulation, analyses, data preparation …
- Clean-up intermediate data
- Store final *results*
Why Scientific Workflows?

General Features

- **Abstraction**, scientists can focus on their research and not computation management
- **Easy composition and execution**
- Enables parallel, distributed **computations**

Different types

- abstract vs. concrete
- task-flow vs. data-flow
- files vs. stream-based

**Workflow Management Systems (WMS)**

Provide tools to generate the scientific workflow
WORKFLOW MANAGEMENT SYSTEMS

**Taverna**
https://taverna.incubator.apache.org

**Kepler**
https://kepler-project.org

**Makeflow**
http://ccl.cse.nd.edu/software/makeflow

**Nextflow**
https://www.nextflow.io

**KNIME**
https://www.knime.org

**VisTrails**
http://vistrails.org

**FireWorks**
https://pythonhosted.org/FireWorks

**Swift**
http://swift-lang.org

**Pegasus**
http://pegasus.isi.edu
BIG DATA FRAMEWORKS

Apache Spark
https://spark.apache.org/

Apache Storm
http://storm.apache.org/

MapReduce
https://hadoop.apache.org/

dispel4py
http://dispel4py.org

Apache Flink
https://www.nextflow.io
Asterism Framework

Easy to understood, platform-independent, open-source

Simplifies the development applications running across multiple heterogeneous

**How ?**

Combining the strengths of

- Traditional WMS
- **Pegasus**

- New stream-based data-flow systems
- **dispel4py**

Ewa Deelman, ISI-USC, US
Rafael Ferreira da Silva, ISI-USC, US

Rosa Filgueira, BGS-NERC, UK
Amrey Krause, EPCC-UoE, UK
Malcolm Atkinson, Informatics-UoE, UK
Alessandro Spinuso, KNMI, Netherlands

Research Visit to ISI-USC, US - 2016
dispel4py parallel stream-based dataflow system

Automation
Automates pipeline executions
Concurrent, distributed computations
Stream-based model

Workflow Composition
Python Library
Groupings
Notebooks-Jupyter

Mapping
Sequential
Multiprocessing
MPI
Apache Storm
Apache Spark (Prototype)

Optimisation
Multiple streams
Avoids I/O operations
Agile Provenance

Key-features: Automatic parallelization/mappings, concurrent & stream-based

https://github.com/dispel4py/dispel4py
Pegasus workflow system

**Automation**
- Automates pipeline executions
- Parallel, distributed computations
- Automatically executes data transfers
- Heterogeneous resources
- Task-oriented model
- Application is seen as a black box

**Debugging**
- Workflow execution and job performance metrics
- Real-time monitoring, graphs, provenance

**Recovery**
- Job failure detection
- Checkpoint Files
- Job Retry
- Rescue DAGs

**Optimisation**
- Job clustering
- Data cleanup

Key-features: Automatic data movement, cleanup, heterogeneous & coordination

LIGO project: First **detection of gravitational waves** from colliding black holes

http://pegasus.isi.edu
PEs represent the basic computational unit
Data transformation, scientific method, service request

PEs are the “Lego bricks” of tasks and users can assemble them into a workflow as they wish

General PE features
Consumes any number and types of input streams
Produce any number and types of output streams
dispel4py basic concepts – Graph

Topology of the workflow: connections between Pes

Users focus on the algorithm to implement or the service to use
dispel4py basic concepts – 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
dispel4py basic concepts – Mappings

**Sequential**
- Sequential mapping for local testing
- Ideal for local resources: laptops/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
Pegasus Architecture

13
Pegasus Environments – Heterogeneous and Homogenous

submit host
(e.g., user’s laptop)

Compute site A

Input data site
Data staging site

Data staging site

Compute site B

Input data site
Output data site

Output data site

shared filesystem

object storage
Complementary systems

Task-flow

Data-flow

Task: Executable (black box)

Processing Elements: python objects
**Pegasus** to distribute and execute each dispel4py workflow

dispel4py to represent different parts of applications

**Pegasus** to distribute and execute each dispel4py workflow
Seismic Ambient Noise Cross-Correlation

IRIS USA TA array - 1000 stations - 4999500 cross-correlations per hour

Preprocesses (Phase 1) and cross-correlates traces (Phase 2) from multiple seismic stations

VERCE project both phases on the same HPC resource: Tested all the mappings and the fastest: MPI
Evaluation- Seismic Ambient Noise Cross-Correlation

Execution among heterogeneous systems

IRIS database (stations)

input data (~150MB)

data transfers between sites performed by Pegasus

Phase 1

Compute site A (MPI-based)

dispel4py

Phase 2

Compute site B (Apache Storm)

dispel4py

submit host (e.g., user’s laptop)

output data (~40GB)
Evaluation - Seismic Ambient Noise Cross-Correlation

IRIS database (stations)

Data transfers between sites performed by Pegasus

Container 1
Submit host (e.g., user’s laptop)

output data (~40GB)

Container 2
Compute site A (MPI-based)

dispel4py
Phase 1

input data (~150MB)

IRIS database (stations)

Compute site B (Apache Storm)

dispel4py
Phase 2

Container 3

Evaluation - Seismic Ambient Noise Cross-Correlation

NSF-Chameleon Cloud Allocation: 40 instances

Container 1
Submit host (e.g., user’s laptop)

output data (~40GB)

https://github.com/dispel4py/pegasus_dispel4py
Reminder

Data-Intensive Application

dispel4py preproc. (Phase 1)

dispel4py proc. (Phase 2)

Container 1

Submit Site

Container 2

MPI Cluster Phase 1

Storm Cluster Phase 2

Container 3

Dockerfiles to configure containers images → Stored in our GitHub → Linked to DockerHub → stored/share/download images
Data-Intensive Application

### Execution Environment -- Container 1: Pegasus, HTCondor, dispel4py

**Pegasus workflow - DAX**

- **f.a**: stations
- **f.b**: preproc. data
- **f.c**: cross-corr. results

**Legend**
- Pegasus task (dispel4py workflow inside)
- Create Directory Job
- Stage-in Job
- Clustering Pegasus Job (dispel4py workflow inside)
- Cleanup Job
- Stage-out Job
- Registration Job

**Execution**
- **Execution environment**
  - MPI Cluster
  - Storm Cluster

**Workflow Mapper**

**Pegasus executable**

**Asterism Seismic Cross Correlation workflow**

**Reminder**

**Dockerfiles** to configure containers images ➔ Stored in our GitHub ➔ Linked to DockerHub ➔ stored/share/download images

**Container 1**
- MPI Cluster
- Submit Asterism

**Container 2**
- Storm Cluster

**Container 3**

1 instance as Container 1
Data-Intensive Application

**Container 1**
Submit Asterism

**Container 2**
MPI Cluster
Phase 1

**Container 3**
Storm Cluster

**Execution environment -- Container 2**
MPI cluster, dispel4py, obspy

**Legend**
- Clustered Pegasus job
- dispel4py PE
- MPI process

**Dockerfiles** to configure containers images → Stored in our GitHub → Linked to DockerHub → stored/share/download images

1 instance as Container 2 (MPI head node)
16 instances as Container 2 (MPI workers)

dispel4py preproc. (Phase 1)

dispel4py proc. (Phase 2)

Reminder
Data-Intensive Application

Container 1
- Submit Asterism
- MPI Cluster
  - Phase 1
- Storm Cluster
  - Phase 2

Container 2
- MPI Cluster
  - Phase 1

Container 3
- Storm, dispel4py, Obspy
- 3 instances as Container 3 (zookeeper, nimbus, Storm UI)
- 16 instances as Container 3 (Supervisors)

dispel4py preproc. (Phase 1)
- Read
- Trace
- Write

dispel4py proc. (Phase 2)
- Read Prep
- xCorr
- Write xCorr.

Dockerfiles to configure containers images ➔ Stored in our GitHub ➔ Linked to DockerHub ➔ stored/share/download images

Reminder
Asterism Evaluations

Experiment 1: Data from IRIS services (394 stations)

**Time**
- Phase 1 – 8 minutes
- Phase 2 – 2 hours
- Moving data < 1 minute

**Data size**
- Input data 150MB
- Output data ~40GB

Experiment 2: Workflow for 3 days requesting data every 2 hours

**Scope of this work**

- Executing & paralyzing automatically data-intensive applications
- in heterogeneous systems with different enactment engines

Maximizing performance
- both phases in the MPI cluster
- increasing the number of Storm Supervisors
DIaaS: Data-Intensive workflow as a services

- Integrated, complete, easy-to-use, portable approach to run data-intensive workflow
  - Science Gateways
- Packed specialized software
- Reduce the time to build such systems
- Containers turned on when they are needed
- Containers are interchangeable
- Images can be extended

ASTERISM framework

DIaaS
Conclusions and Future works

Asterism: Easy-to-use system to empower data-driven research

New framework that automatically
- manage the entire workflow, monitor its execution
- handle data transfers between different platforms
- map to different enactment engines at runtime

DIaaS: Data Intensive Workflows as Service

Easy composition & deployment of data-intensive workflows

Real domain application on the NSF-Chameleon cloud

Future works

More e-Infrastructures and mappings

Asterism via management tool
Recent developments -> Singularity + BGS
Resources

Containers

Container 1: Pegasus + dispel4py + HTCondor – https://github.com/dispel4py/pegasus_dispel4py

Container 2: MPI cluster – https://github.com/dispel4py/pegasus_dispel4py/tree/master/Docker


DockerHub image: https://hub.docker.com/r/dispel4py/pegasus

Script for the deployment of the Docker Containers: https://github.com/dispel4py/pegasus_dispel4py/blob/master/commands_to_deploy_environment.sh
Asterism: Pegasus and dispel4py hybrid workflows for data-intensive science

Thank You

Questions?

Rosa Filgueira, Ph.D.
rosa@bgs.ac.uk
http://www.rosafilgueira.com
https://github.com/rosafilgueira

Amy Krause, PhD
a.krause@epcc.ed.ac.uk

Relevant Information

Pegasus Website
http://pegasus.isi.edu

dispel4py GitHub
https://github.com/dispel4py/

Research Object
https://scitech.isi.edu/ro/asterism/

Workflows and Training Material
https://github.com/rosafilgueira/dispel4py_workflows
https://github.com/rosafilgueira/dispel4py_training_material