Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

EXCELENCIA
SEVERO
OCHOA

Programming Distributed Computing Platforms with COMPSs

Javier Alvarez, Rosa M. Badia, Javier Conejero, Jorge Ejarque, Daniele Lezzi, Francesc Lordan, Nihad Mammadli, Cristian Ramon-Cortes, Salvi Solà

Workflows & Distributed Computing Group

28-29/01/2020

Barcelona

Outline

Day 1

- Roundtable (9:30 – 10:00): Presentation and background of participants
- Session 1 (10:00 – 10:30): Introduction to COMPSs
  - Motivation
  - Setup of tutorial environment
- Session 2 (10:30-11:15): PyCOMPSs: Writing Python applications
- Coffee break (11:15 – 11:45)
- Session 3 (11:45 a 13.00) Python Hands-on using Jupyter notebooks
- Lunch break (13:00-14:30)
- Session 4 (14:30 - 15:00) Machine learning with dislib
- Session 5 (15:00 -16:30): Hands-on with dislib
- SLIDES
  - http://compss.bsc.es/releases/tutorials/tutorial-PATC_2020/

Outline

Day 2

- Session 6 (9:30-11:00): Java & C++
  - Writing Java applications
  - Java Hands-on + debug
  - C++ Syntax
- Coffee break (11:00 – 11:30)
- Session 7 (11:30-13:00): COMPSs Advanced Features
  - Using binaries and MPI code, Fault Tolerance and Exception management, Numba
  - COMPSs execution environment
- Lunch break (13:00 – 14:30)
- Session 8 (14:30-16:30): Cluster Hands-on (MareNostrum)
- COMPSs Installation & Final Notes

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

INTRODUCTION

Motivation

- New complex architectures constantly emerging
  - With their own way of programming them
    - Fine grain: e.g. Programming models and APIs to run with GPUs, NVMs (Non-Volatile Memories)
    - Coarse grain: e.g. APIs to deploy in Clouds
  - **Difficult** for programmers
    - Higher learning curve / Time To Market (TTM)
    - What about non computer scientists???
  - **Difficult** to understand what is going on during execution
    - Was it fast? Could it be even faster? Am I paying more than I should? (**Efficiency**)
  - Tune your application for each architecture (or cluster)
    - E.g. partitioning data among nodes

Motivation

- Create tools that make developers’ life **easier**
  - Allow developers to focus on their problem
  - Intermediate layer: let the difficult parts to those tools
    - Act on behalf of the user
    - Distribute the work through resources
    - Deal with architecture specifics
    - Automatically improve performance
  - Tools for visualization
    - Monitoring
    - Performance analysis

BSC vision on programming models

Applications

PM: High-level, clean, abstract interface

Power to the runtime

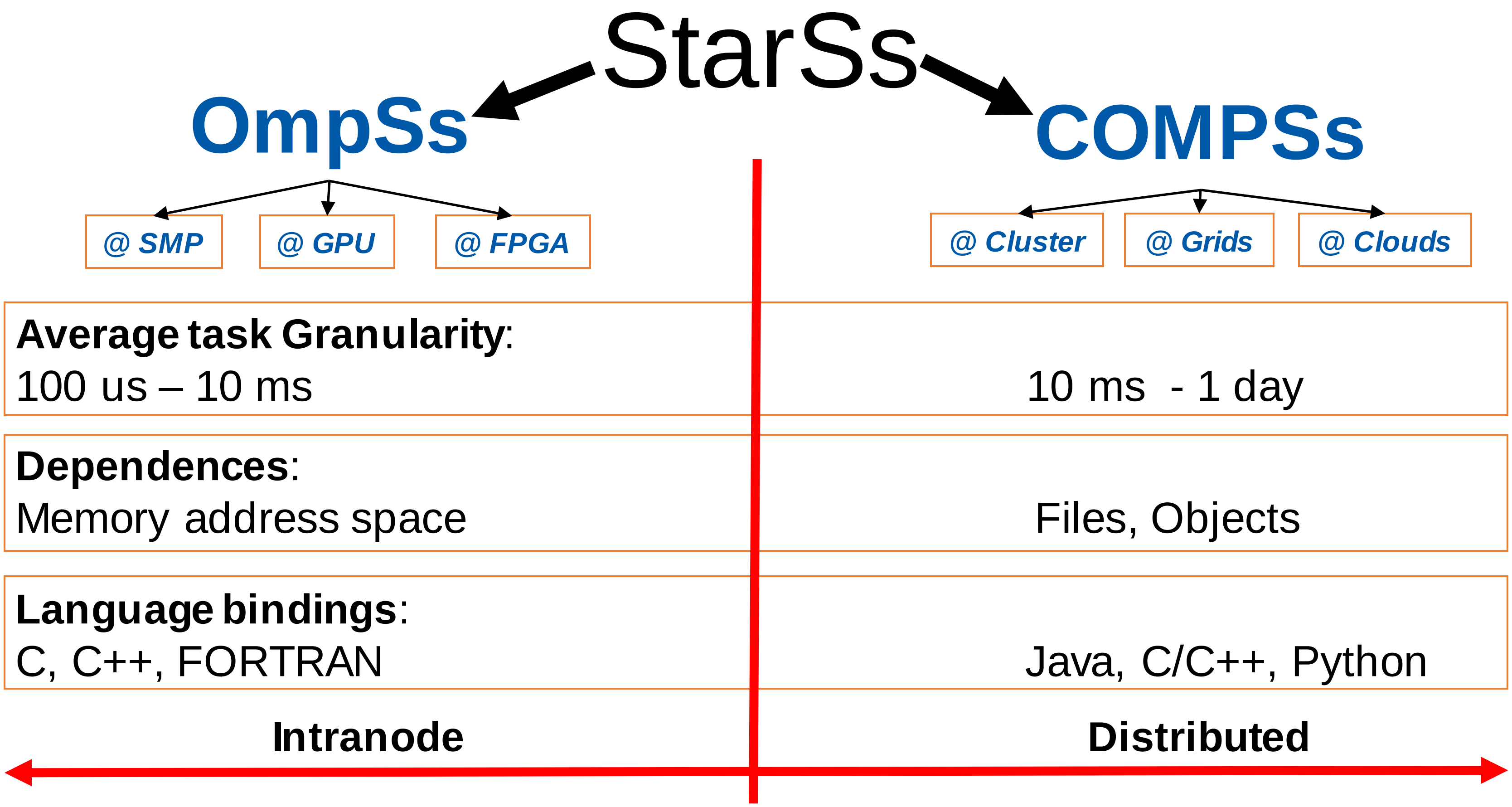
API

Program logic independent of computing platform

General purpose
Task based
Single address space

Intelligent runtime, parallelization, distribution, interoperability

BSC vision on programming models



StarSs

OmpSs

@ SMP

@ GPU

@ FPGA

COMPSs

@ Cluster

@ Grids

@ Clouds

Average task Granularity:

100 us – 10 ms

10 ms - 1 day

Dependences:

Memory address space

Files, Objects

Language bindings:

C, C++, FORTRAN

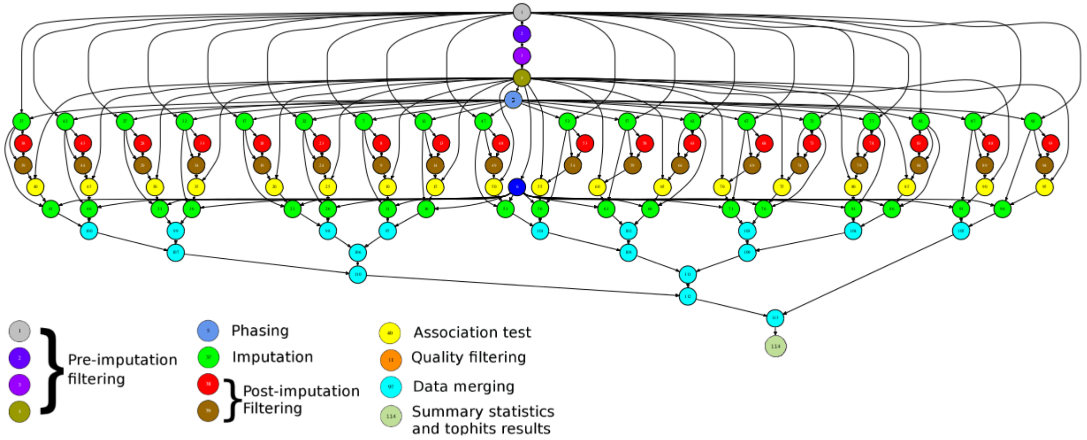
Java, C/C++, Python

Intranode

Distributed

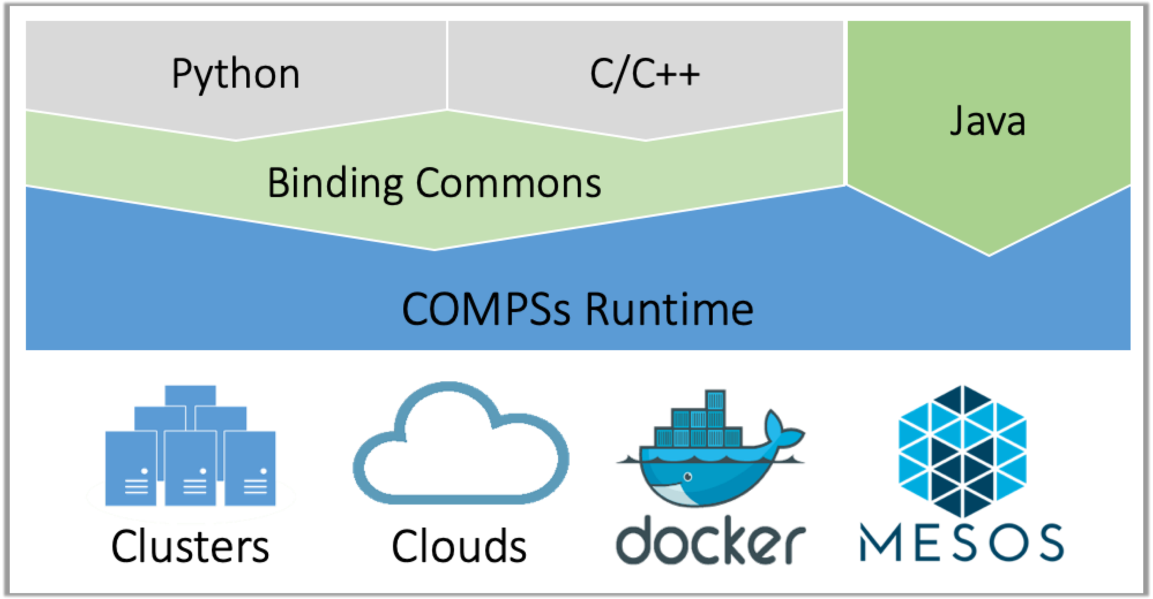
Programming with COMPSs

- Sequential programming
- General purpose programming language + annotations/hints
  - To identify tasks and directionality of data
- Task based: task is the unit of work
- Simple linear address space
- Builds a task graph at runtime that express potential concurrency
  - Implicit workflow
- Exploitation of parallelism
  - … and of distant parallelism
- Agnostic of computing platform
  - Enabled by the runtime for clusters, clouds and grids



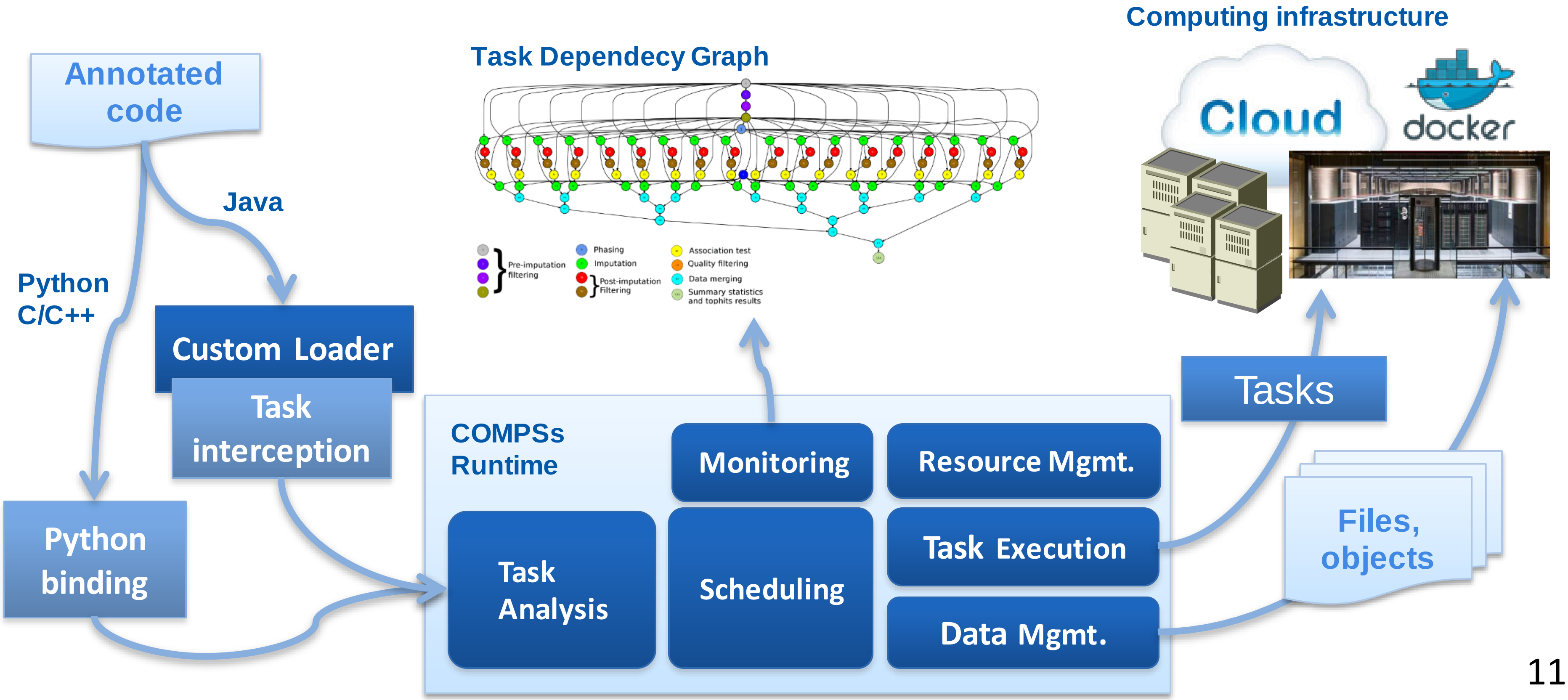
Programming with COMPSs

- Support for other types of parallelism
  - Threaded tasks (I.e., MKL kernels)
  - MPI applications -> tasks that involve several nodes
  - Integration with BSC OmpSs
- Available in MareNostrum, in the EGI Federated Cloud and in Chameleon Cloud



COMPSs runtime

- PyCOMPSs/COMPSs applications executed in distributed mode following the master-worker paradigm
- Sequential execution starts in master node
- Tasks are offloaded to worker nodes
- All data scheduling decisions and data transfers are performed by the runtime



Annotated code

Java

Python C/C++

Custom Loader

Task interception

Python binding

Task Dependecy Graph

Computing infrastructure

Tasks

Files, objects

COMPSs Runtime

Monitoring

Resource Mgmt.

Task Analysis

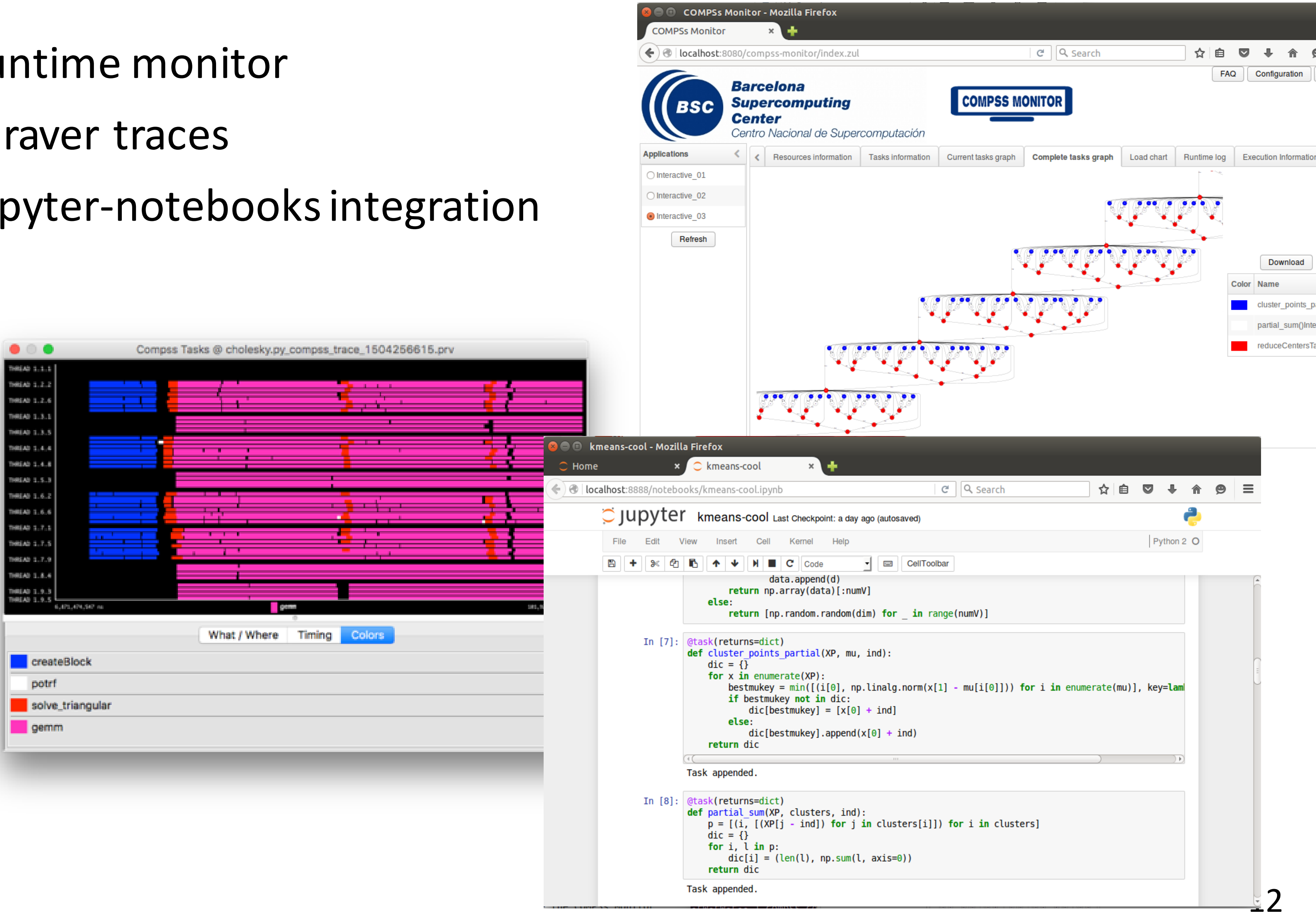
Scheduling

Task Execution

Data Mgmt.

PyCOMPSs development environment

- Runtime monitor
- Paraver traces
- Jupyter-notebooks integration



```
            data.append(d)
        return np.array(data)[:numV]
    else:
        return [np.random.random(dim) for _ in range(numV)]
```

```
In [7]: @task(returns=dict)
        def cluster_points_partial(XP, mu, ind):
            dic = {}
            for x in enumerate(XP):
                bestmukey = min([(i[0], np.linalg.norm(x[1] - mu[i[0]])) for i in enumerate(mu)], key=lam
                if bestmukey not in dic:
                    dic[bestmukey] = [x[0] + ind]
                else:
                    dic[bestmukey].append(x[0] + ind)
            return dic

        Task appended.
```

```
In [8]: @task(returns=dict)
        def partial_sum(XP, clusters, ind):
            p = [(i, [(XP[j - ind]) for j in clusters[i]]) for i in clusters]
            dic = {}
            for i, l in p:
                dic[i] = (len(l), np.sum(l, axis=0))
            return dic

        Task appended.
```

Projects where COMPSs is used/developed

bioexcel
Center of Excellence for Computational Biomolecular Research

mF2C

CLASS

ExaQUte
Exascale Quantification of Uncertainties for Technology and Science Simulation

ELASTIC

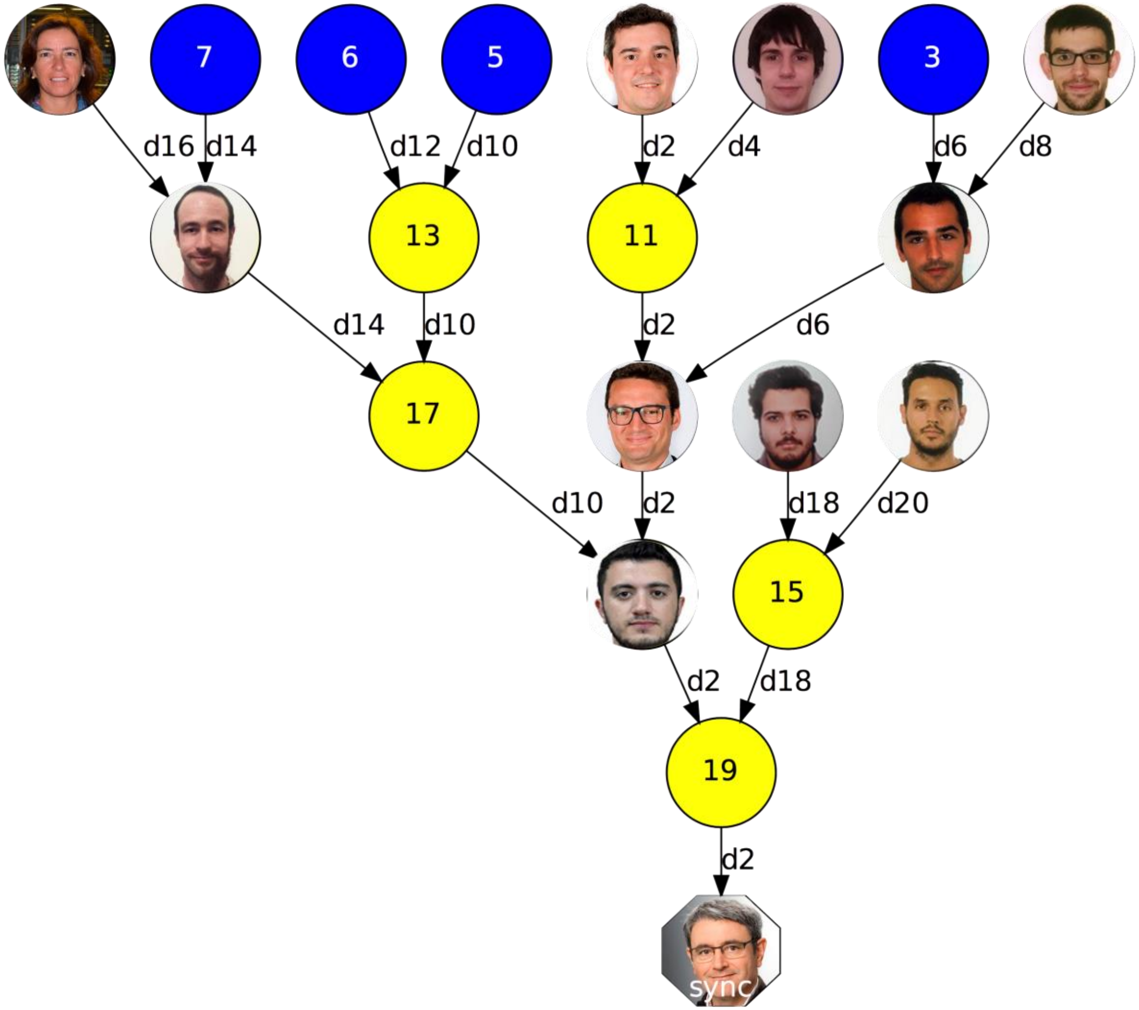
LAND SUPPORT

EXPERTISE

FUJITSU

Joint Laboratory for Extreme-Scale Computing
Inria
NCSA
Argonne NATIONAL LABORATORY
BSC Barcelona Supercomputing Center
JÜLICH FORSCHUNGSZENTRUM
RIKEN

The WDC team



http://compss.bsc.es

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

SETUP OF THE TUTORIAL ENVIRONMENT

Setup

- From Linux or Mac:
  - https://pypi.org/project/pycompss-player/#quickstart

1. Install docker
2. Install the PyCOMPS player for Docker:

   ```
   sudo python3 -m pip install pycompss-player
   ```
3. Optional (to reduce wait times)

   ```
   docker pull compss/compss-tutorial:2.6
   ```

Setup

- For windows
  - https://pypi.org/project/pycompss-player/#quickstart

1. Download and Install Oracle VirtualBox
   https://www.virtualbox.org/

2. Download the tutorial VM.
   http://compss.bsc.es/releases/vms/COMPSs-2.6-tutorial.ova

3. Open VirtualBox and import the ova.
   Optional (but recommended to avoid large waiting times)

4. Start de VM, log in (password is compss2019) and run:
   docker pull compss/compss-tutorial:2.6

   Note: If the docker pull command fails be sure you have internet connection, the Docker service is running (sudo service docker start) and your user is in the docker group (sudo usermod -aG *docker $USER)*

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Start PyCOMPSs player

- Open a terminal in your linux/mac laptop or in the VM machine
- Get the tutorial examples:

```
git clone https://github.com/bsc-wdc/tutorial_apps.git
```

- Start PyCOMPss player with the tutorial's image:

```
pycompss init -i compss/compss-tutorial:2.6
```

- Start COMPSs monitor

```
pycompss monitor start
```

- Open browser with URL: http://127.0.0.1:8080/compss-monitor

- Start Jupyter notebook with tutorial apps

```
cd tutorial_apps/python
pycompss jupyter ./notebooks
```

- Open browser with URL: http://127.0.0.1:8888/

  or http://localhost:8888/

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación