



**Barcelona  
Supercomputing  
Center**  
*Centro Nacional de Supercomputación*



# Programming Distributed Computing Platforms with COMPSs

Rosa M. Badia, Javier Conejero, Jorge Ejarque, Daniele  
Lezzi, Francesc Lordan, Nihad Mammadli

Workflows & Distributed Computing Group

25-26/01/2022

Barcelona

# Outline

## Day 1

- Roundtable (9:30 – 10:00): Presentation and background of participants
- Session 1 (10:00 – 10:30): Introduction to COMPSs
  - Motivation
- 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\\_2022/](http://compss.bsc.es/releases/tutorials/tutorial-PATC_2022/)

# 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)
- Session 9 (16:30-16:45) COMPSs Installation & Final Notes



**Barcelona  
Supercomputing  
Center**  
*Centro Nacional de Supercomputación*

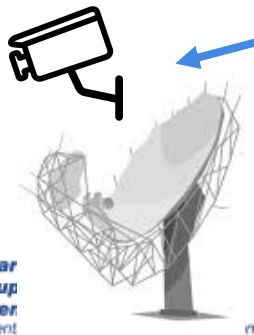
# INTRODUCTION

# 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

- Resources that appear and disappear
  - How to dynamically add/remove nodes to the infrastructure
- Heterogeneity
  - Different HW characteristics (performance, memory, etc)
  - Different architectures -> compilation issues
- Network
  - Different types of networks
  - Instability
- Trust and Security
- Power constraints from the devices in the edge
- Data & Storage



Sensors  
Instruments  
Actuators



Edge devices



Fog devices

AI everywhere



HPC  
Exascale computing  
Cloud

# 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
  - Integration of computational workloads, with machine learning and data analytics

# BSC vision on programming models

Applications

Program logic independent of computing platform

PM: High-level, clean, abstract interface

General purpose  
Task based  
Single address space

Power to the runtime

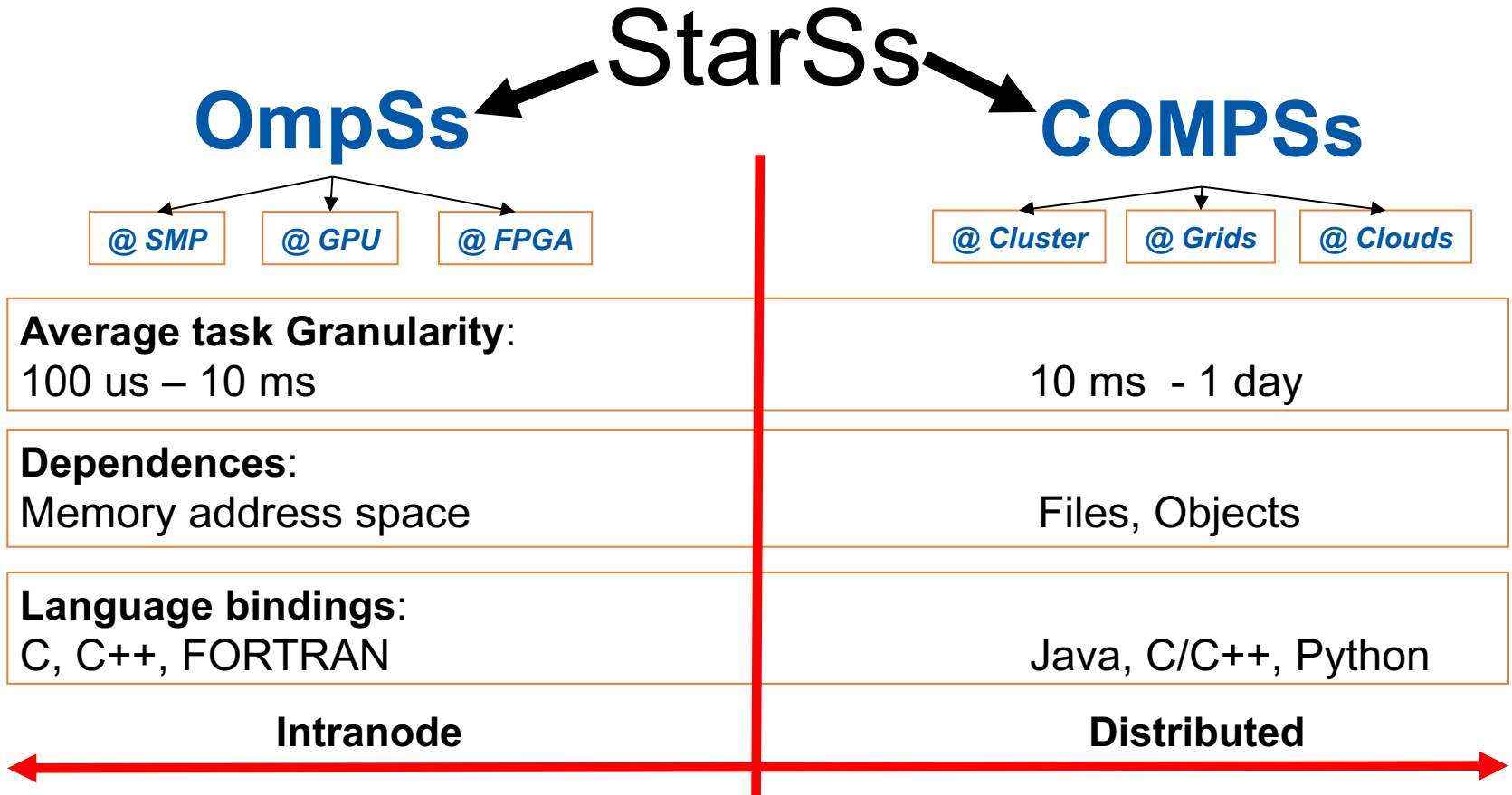
Intelligent runtime, parallelization, distribution, interoperability

API



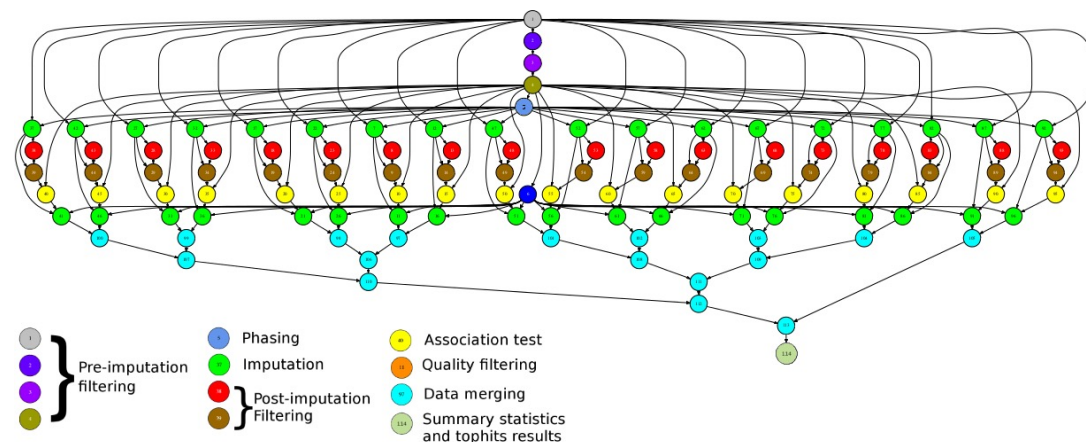


# BSC vision on programming models



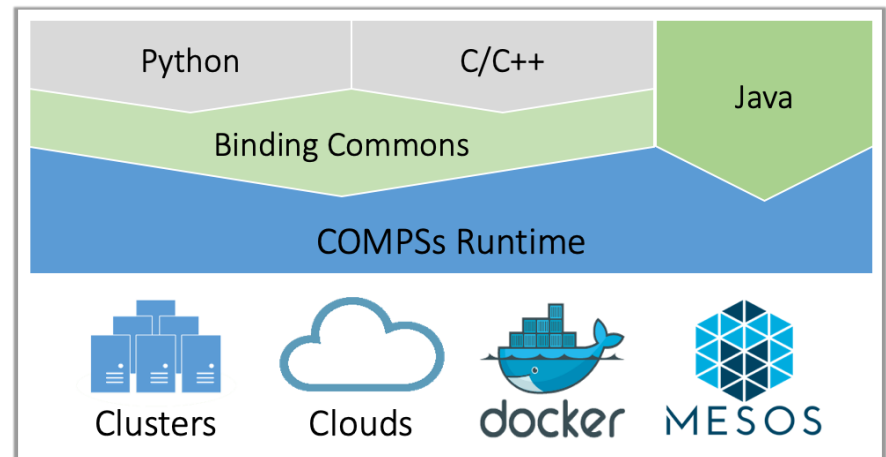
# Programming with COMPSs

- 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



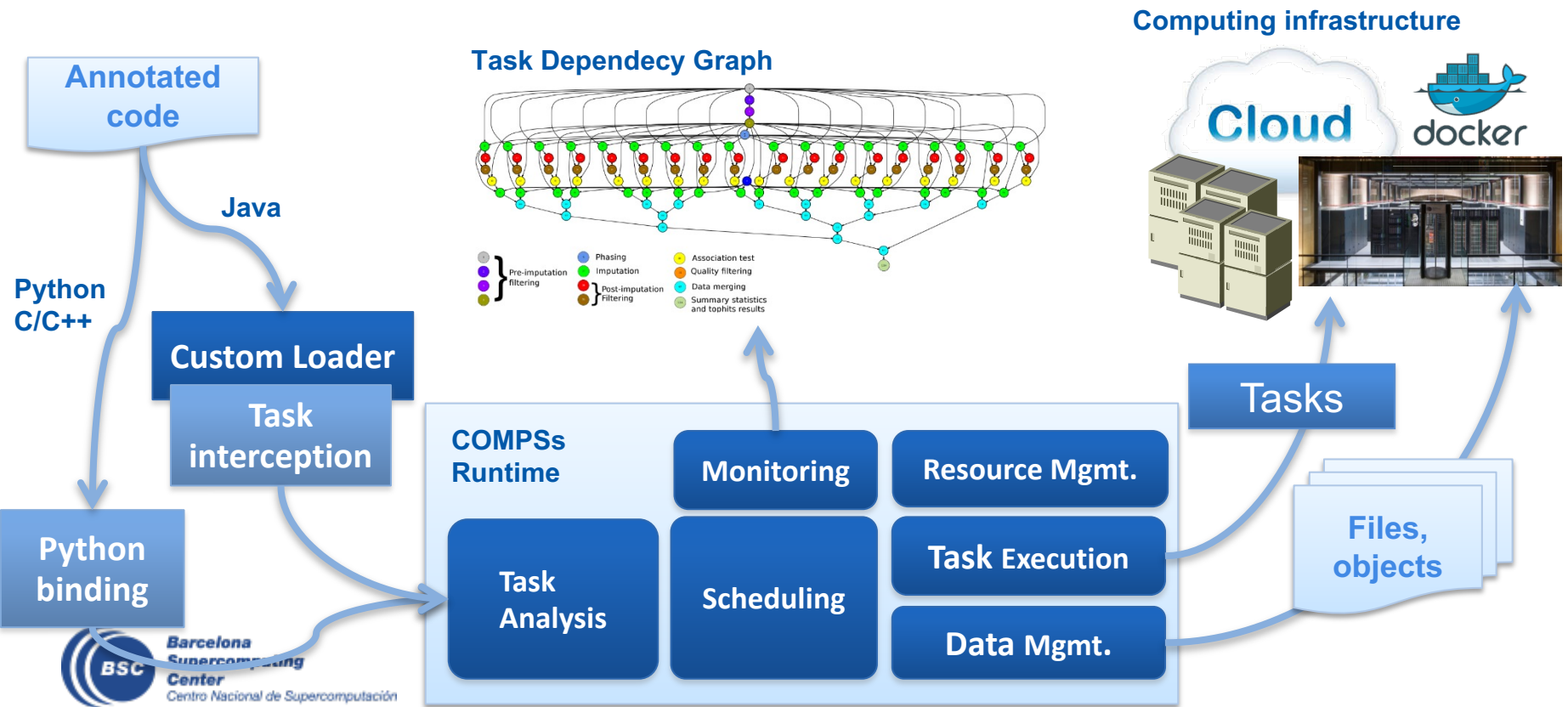
# Programming with COMPSs

- Support for other types of parallelism
  - Threaded tasks (I.e., MKL kernels)
  - MPI applications -> tasks that involve several nodes
  - Integration with BSC OmpSs
  - Streaming tasks for data flow executions
- Support to Failure Management
- Parallel Machine Learning with dislib
- Available in MareNostrum and other supercomputers in Europe, in the EGI Federated Cloud and in Chameleon Cloud



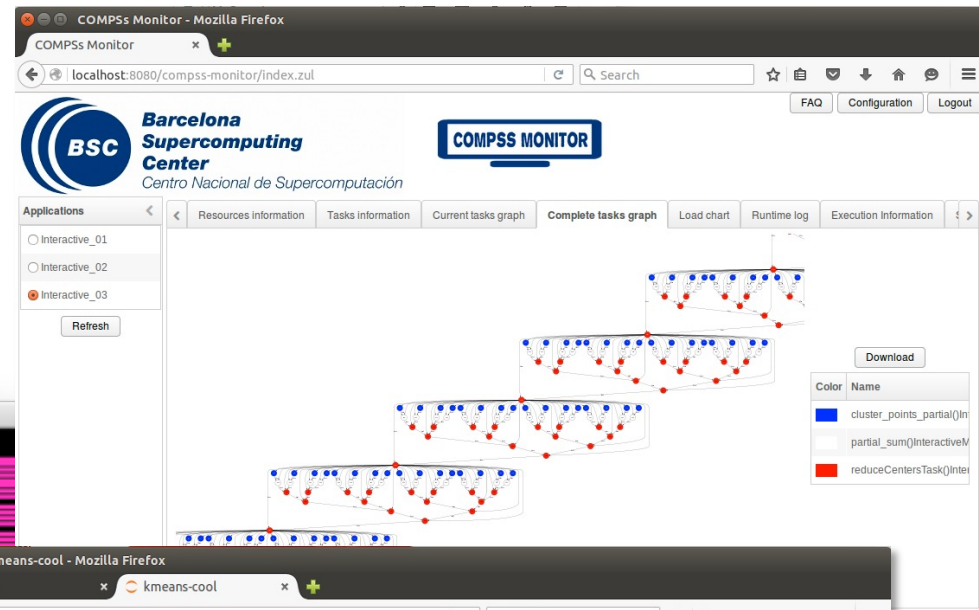
# COMPSs runtime

- 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



# PyCOMPSs development environment

- Runtime monitor
- Paraver traces
- Jupyter-notebooks integration



kmeans-cool - Mozilla Firefox

localhost:8888/notebooks/kmeans-cool.ipynb

Jupyter kmeans-cool Last Checkpoint: a day ago (autosaved)

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

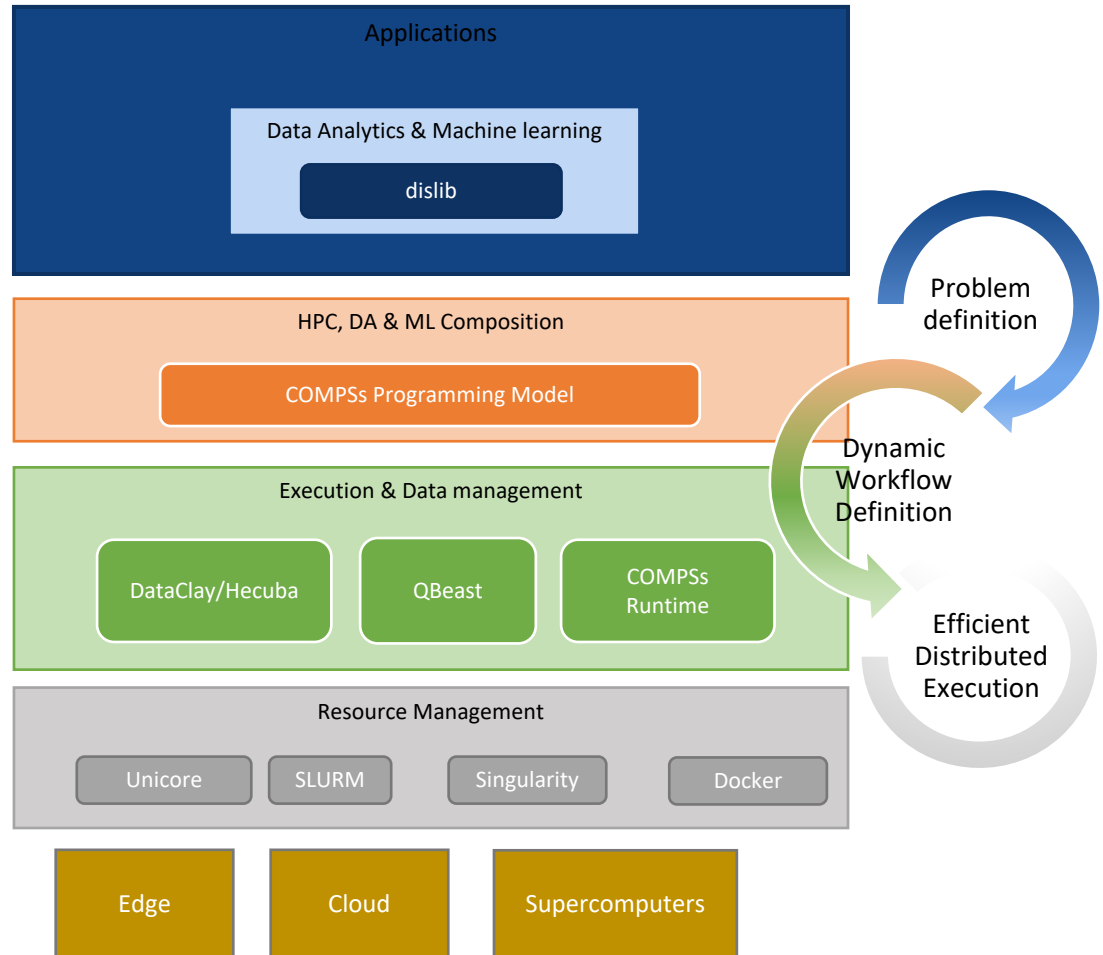
In [7]: @task(returns=dict)
def cluster_points_partial(XP, mu, ind):
    dic = {}
    for x in enumerate(XP):
        bestmukey = min([(i[0], np.linalg.norm(x[1] - mu[i[0]])) for i in enumerate(mu)], key=lambda
        if bestmukey not in dic:
            dic[bestmukey] = [x[0] + ind]
        else:
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

The figure shows a Jupyter Notebook interface. The top part is a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help' menus. The main content area displays Python code for two functions: 'cluster\_points\_partial' and 'partial\_sum'. The code is executed in a cell, and the output shows 'Task appended.' for both functions.

# Conclusions

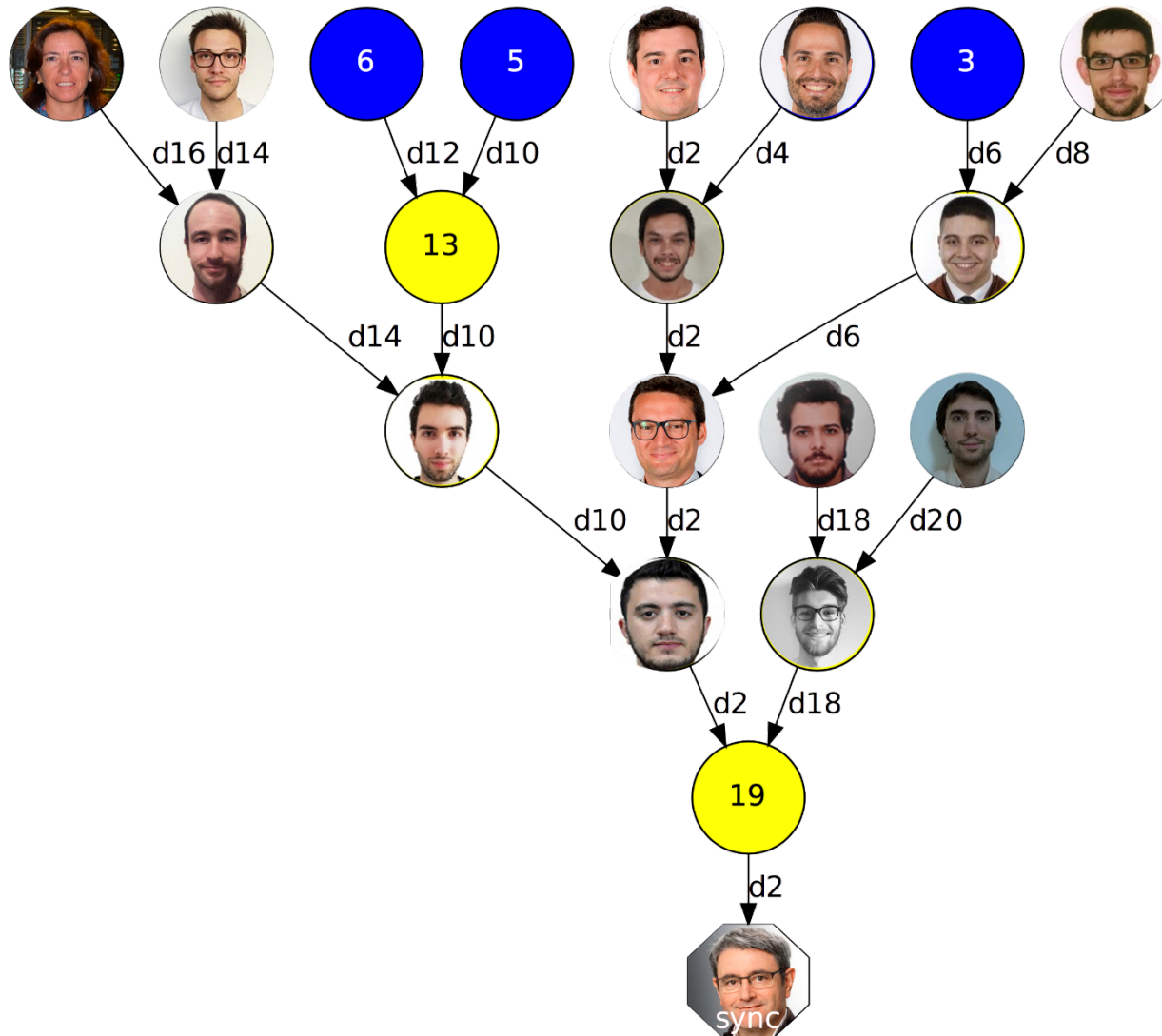
- COMPSs provides a workflow environment that enables the integration of HPC simulation and modelling with big data analytics and machine learning
- Support for dynamic workflows that can change their behaviour during the execution
- Support for dynamic resource management depending on the actual workload needs
- Support for data-streaming enabling the combination of task-flow and data-flow in the same workflow
- Support for persistent storage beyond traditional file systems.



# Projects where COMPSs is used/developed



# The WDC team



<http://compss.bsc.es>