Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

# Programming Distributed Computing Platforms with COMPSs

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

Workflows & Distributed Computing Group

26-27/01/2021

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 - 15:15) Machine learning with dislib
- Session 5 (15:15 -16:30): Hands-on with dislib
- SLIDES
  - http://compss.bsc.es/releases/tutorials/tutorial-PATC_2021/

# 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


# 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

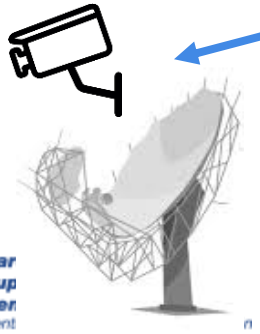
- Resources that appear and disappear
  - How to dynamically add/remove nodes to the infrastructure
- Heterogeneity
  - Different HW characteristics (performance, memory, etc)
  - Different architectures -> compilation issues
- Network
  - Different types of networks
  - Instability
- Trust and Security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

AI everywhere

HPC
Exascale computing
Cloud

Fog devices


Edge devices



Sensors
Instruments
Actuators

# 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

PM: High-level, clean, abstract interface

Power to the runtime

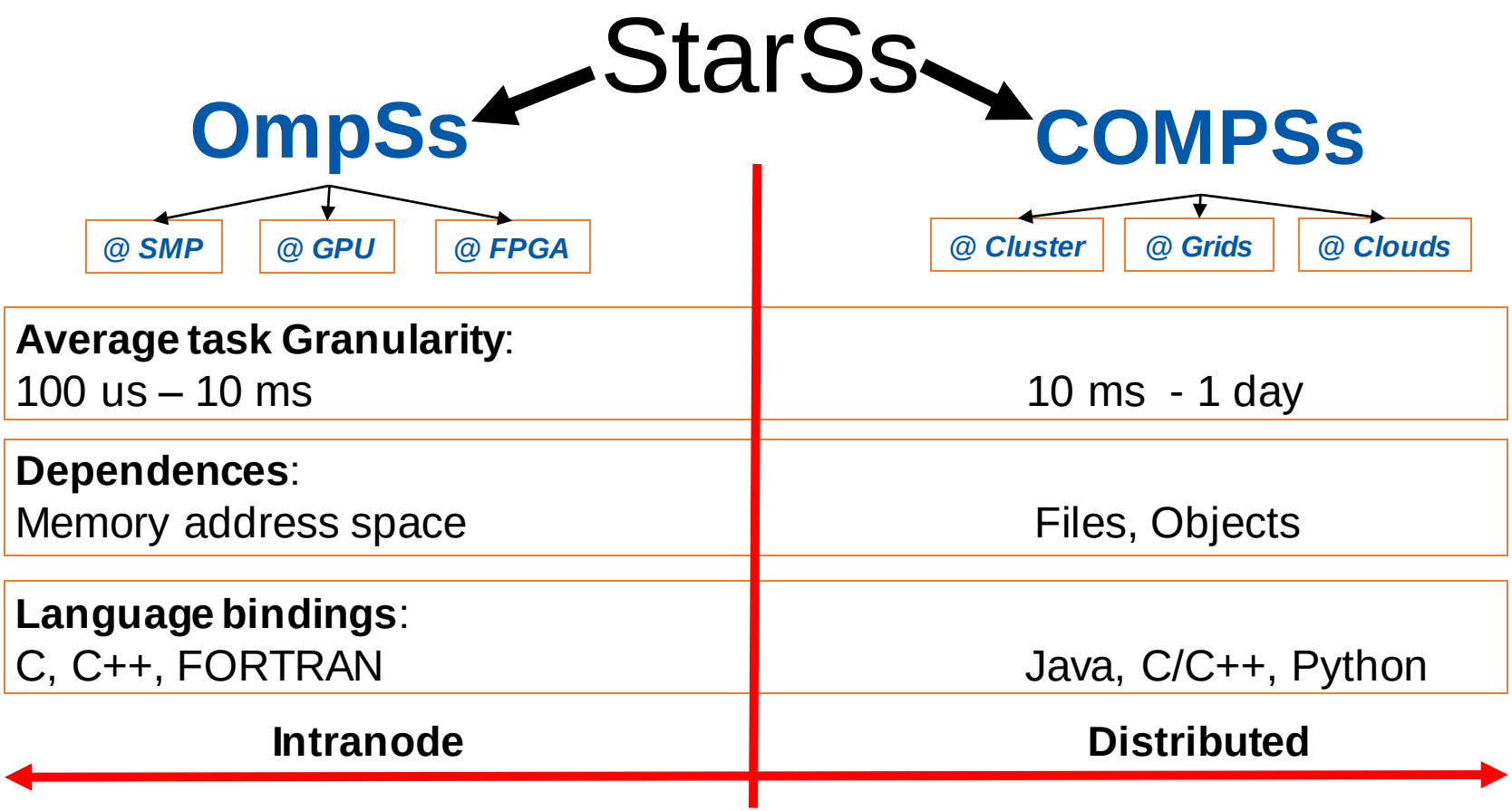
API

Program logic independent of computing platform

General purpose
Task based
Single address space

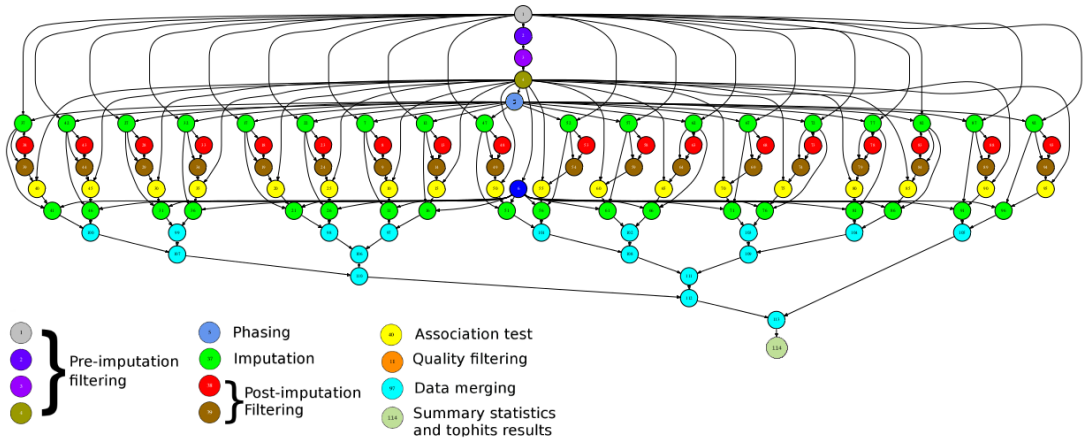
Intelligent runtime, parallelization, distribution, interoperability

# BSC vision on programming models



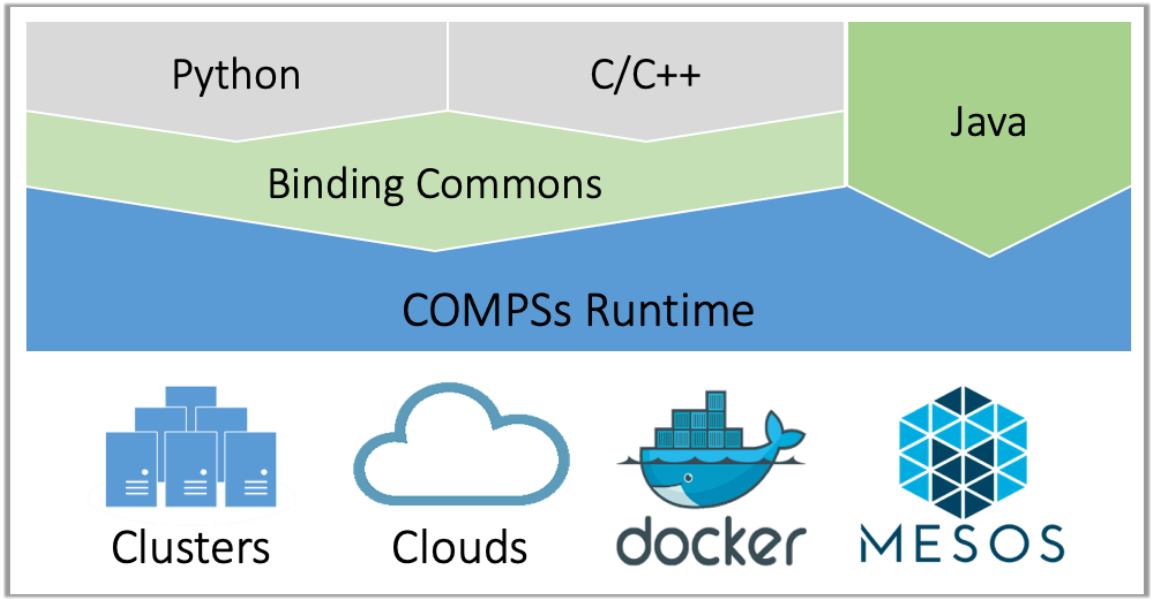
# Programming with COMPSs

- 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



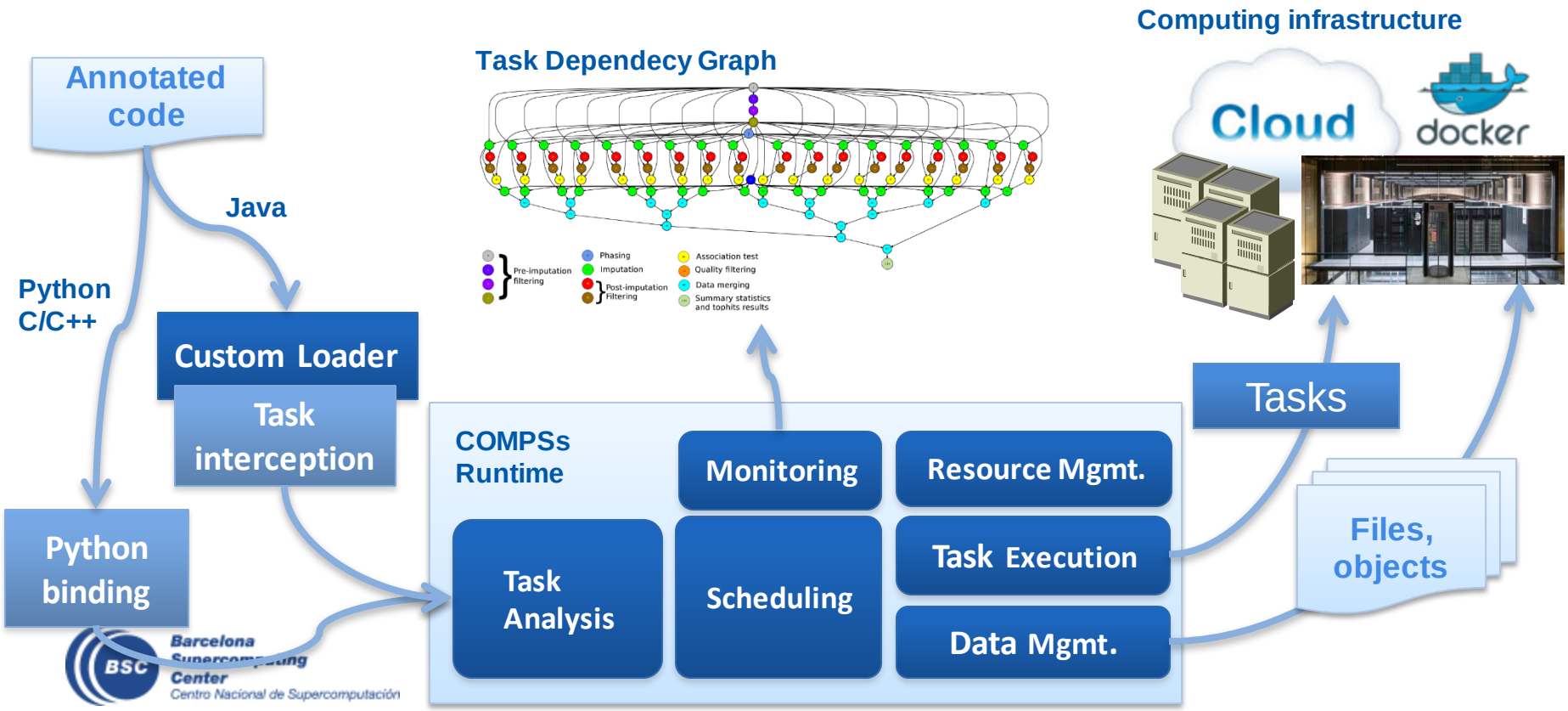
# Programming with COMPSs

- 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



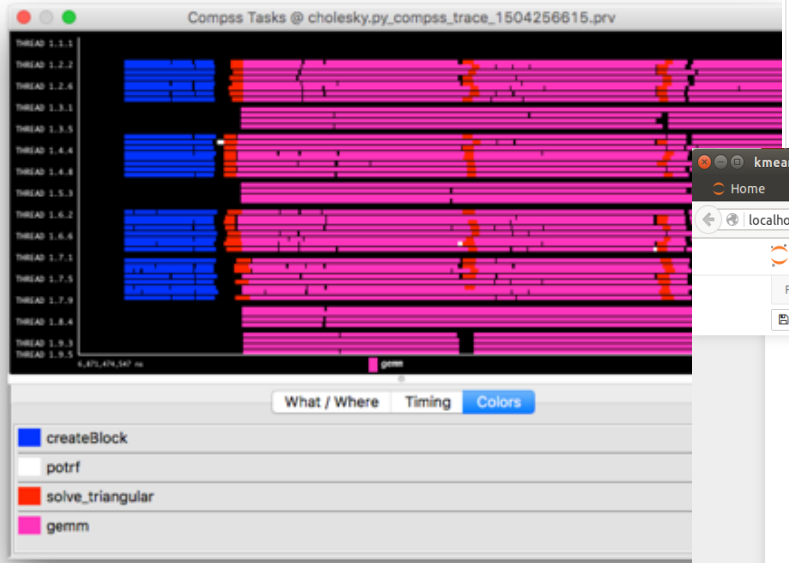
# COMPSs runtime

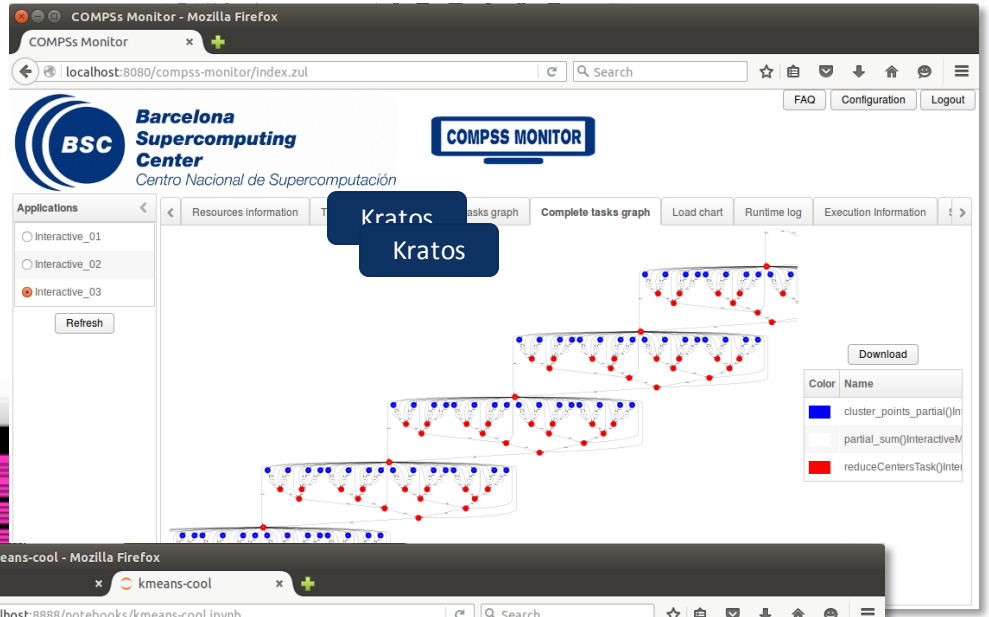
- PyCOMPSs/COMPSs applications executed in distributed mode following the master-worker paradigm
- Sequential execution starts in master node
- Tasks are offloaded to worker nodes
- All data scheduling decisions and data transfers are performed by the runtime

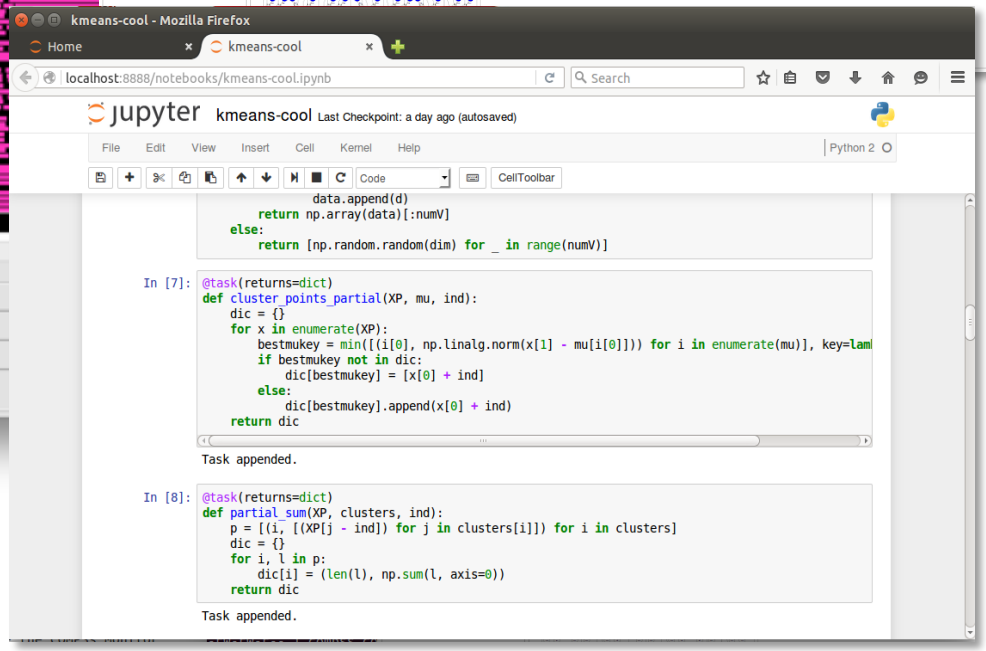


# PyCOMPSs development environment

- Runtime monitor
- Paraver traces
- Jupyter-notebooks integration

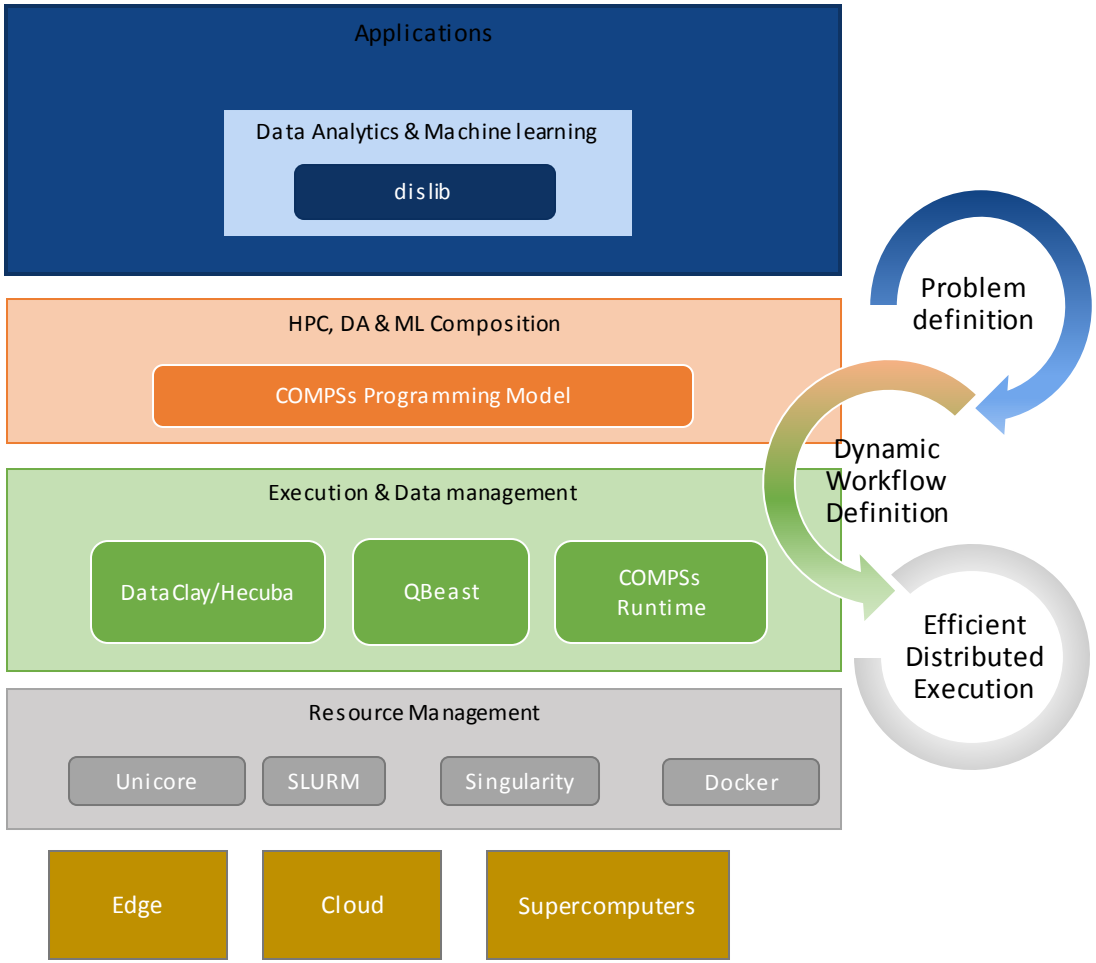


# Conclusions

- 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.



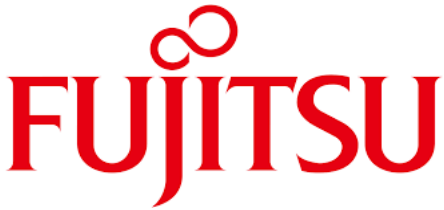
# Projects where COMPSs is used/developed





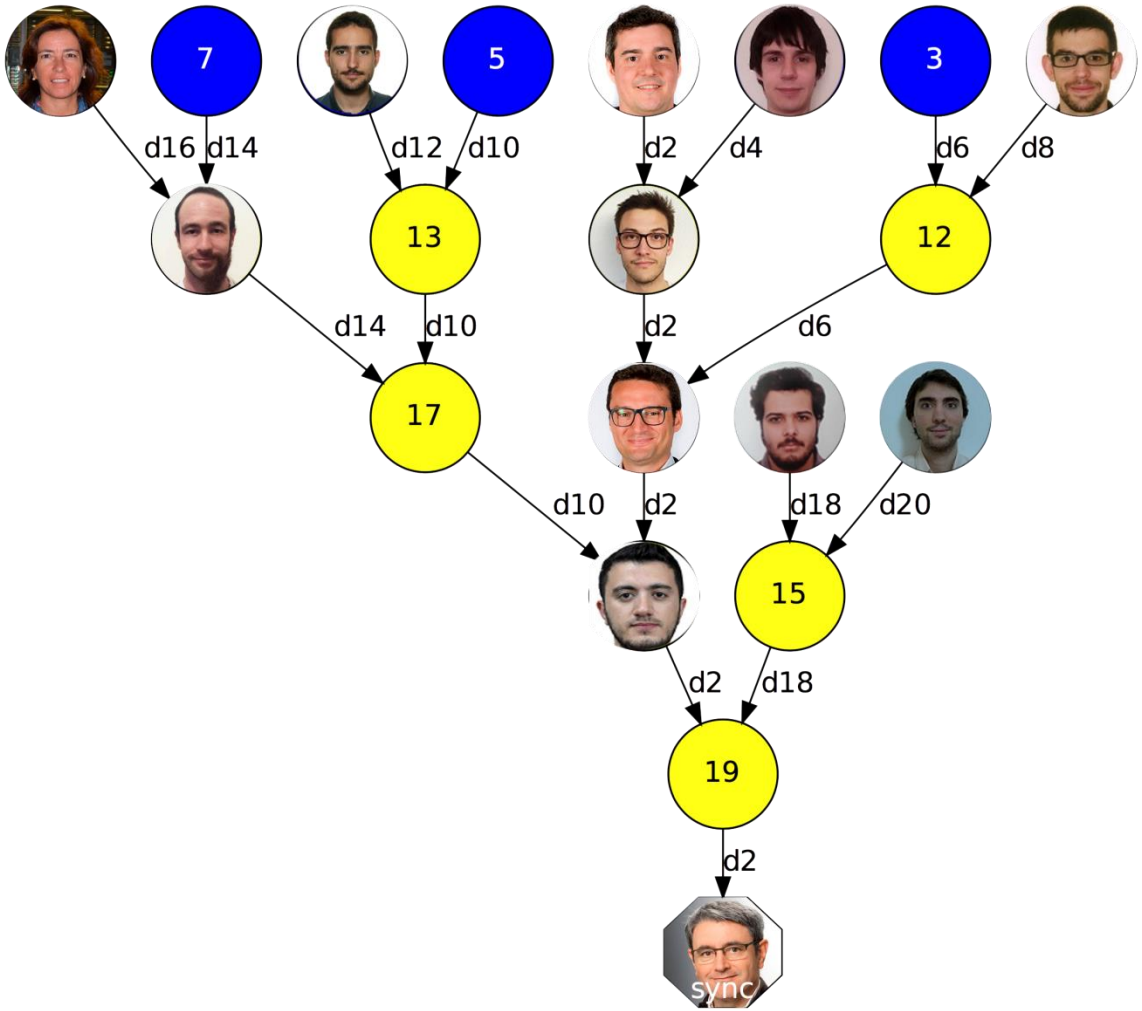








# The WDC team




http://compss.bsc.es


# SETUP OF THE TUTORIAL ENVIRONMENT

# Setup

Linux or Mac:

1. Install docker:
   - Linux: "apt-get install docker" (depends on your distribution)
   - Mac-os: direct download from docker.com. You can find instructions here: https://docs.docker.com/docker-for-mac/install/
2. Get COMPSs docker image:
   - docker pull compss/compss-tutorial:2.8
3. Install pycompss-player:
   - Linux: sudo python3 –m pip install pycompss-player
   - Mac-os: pip install pycompss-player

For Java Hands-on

4. Install maven
   - https://maven.apache.org/install.html
5. A Java IDE is recommended for editing Java code (such as Eclipse)
   - https://www.eclipse.org/downloads/

# 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-2.8.ova

3. Start the VM image
   - Start Virtualbox
   - Import the COMPSs VM image
   - Start COMPSs VM image
     - user: compss
     - password: compss2021
4. Get COMPSs docker image:
   - docker pull compss/compss-tutorial:2.8
     *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)*

# 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:2.8

- 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/