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, Raül Sirvent, Cristian Tatu, Fernando Vazquez

Workflows & Distributed Computing Group

30-31/01/2024

Barcelona

Outline

Day 1

- Roundtable (9:30 – 10:00): Welcome and round table
- Session 1 (10:00 – 10:30): Introduction to COMPSs
- 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_2024/

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Outline

Day 2

- Session 6 (9:30-10:15): Java & C++
  - Writing Java applications
  - Java Hands-on + debug
  - C++ Syntax
- Session 7: (10:15-10:45) Cluster Hands-on (MareNostrum) (Settings)
- Coffee break (10:45 – 11:15)
- Session 8 (11:15-13:00): Cluster Hands-on (MareNostrum)
- Lunch break (13:00 – 14:30)
- Session 9 (14:30-15:30): Provenance with PyCOMPSs (hands-on included)
- Session 10 (15:30-16:30): Running COMPSs with containers (Demo/hands-on included)
- Session 11 (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

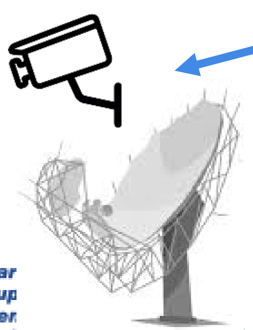
- Resources that appear and disappear
  - How to dynamically add/remove nodes to the infrastructure
- Heterogeneity
  - Different HW characteristics (performance, memory, etc)
  - Different architectures -> compilation issues
- Network
  - Different types of networks
  - Instability
- Trust and Security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

HPC
Exascale computing
Cloud

AI everywhere

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

StarSs

OmpSs

@ SMP | @ GPU | @ FPGA

COMPSs

@ Cluster | @ Grids | @ Clouds

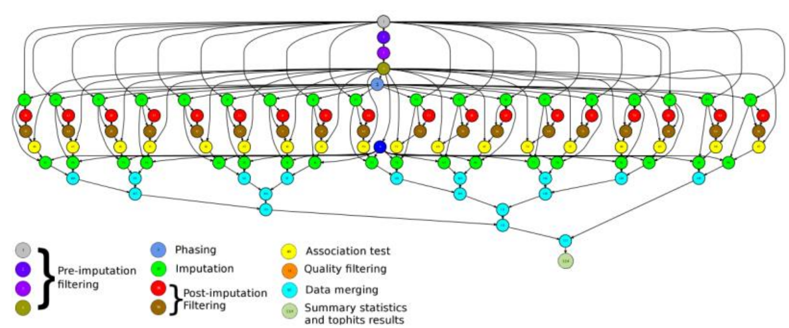
Average task Granularity: 100 us – 10 ms	10 ms - 1 day
Dependences: Memory address space	Files, Objects
Language bindings: C, C++, FORTRAN	Java, C/C++, Python
Intranode	**Distributed**

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

```
@task(c=INOUT)
def multiply(a, b, c):
      c += a*b
```

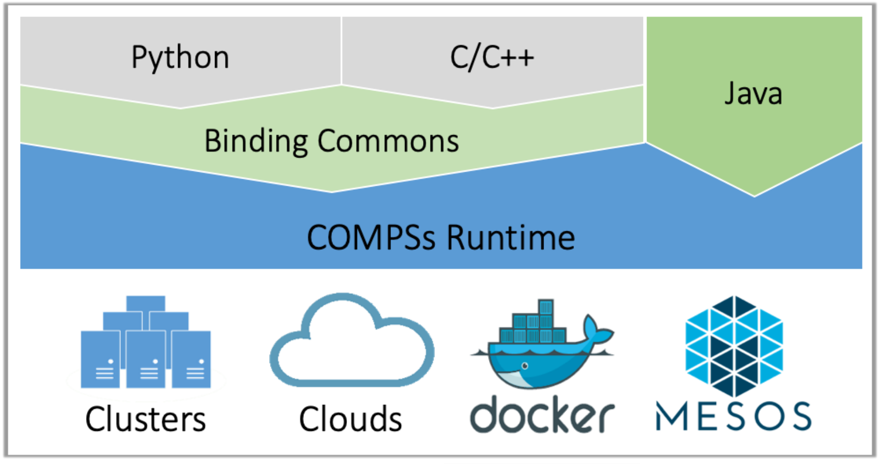
```
initialize_variables()
startMulTime = time.time()
for i in range(MSIZE):
   for j in range(MSIZE):
      for k in range(MSIZE):
         multiply (A[i][k], B[k][j], C[i][j])
compss_barrier()
mulTime = time.time() - startMulTime
```



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Programming with COMPSs

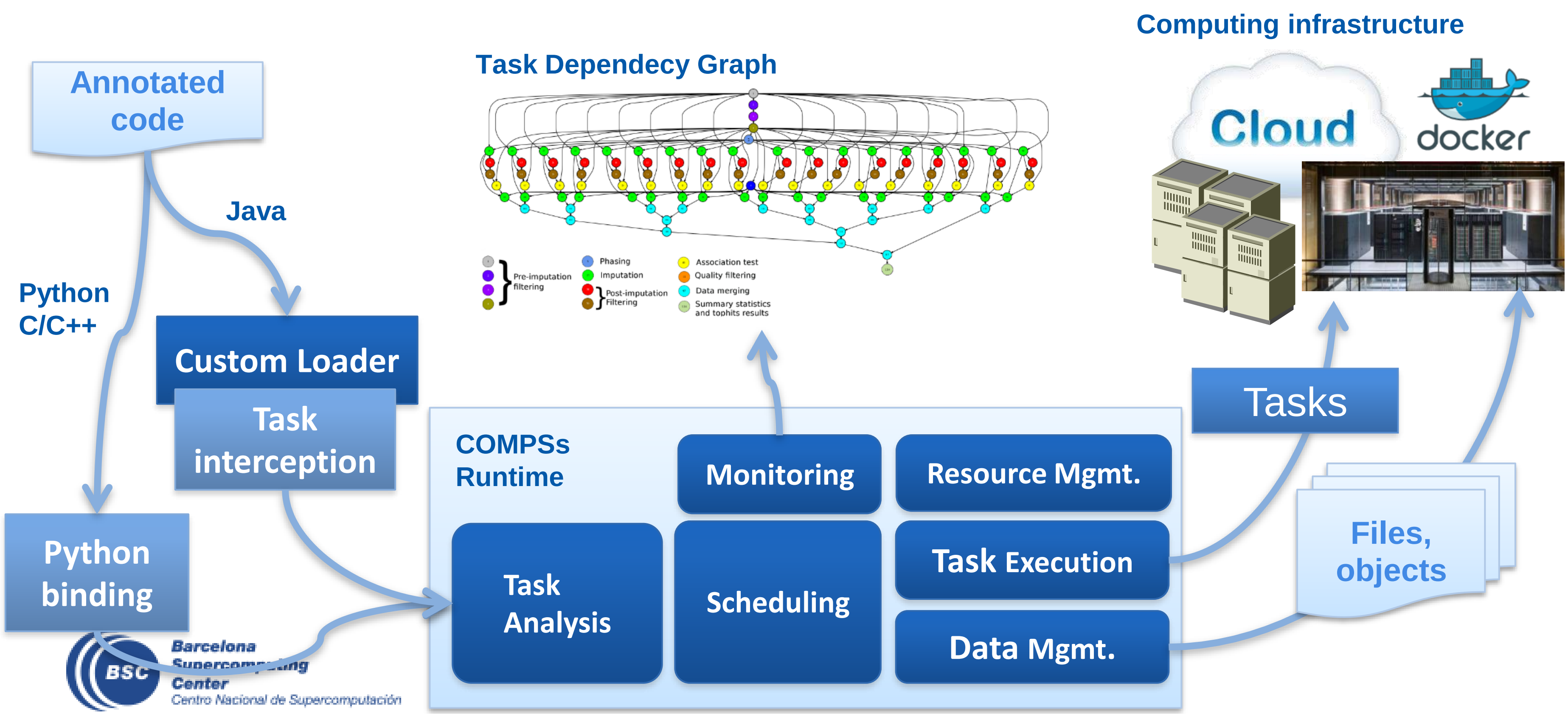
- Support for other types of parallelism
  - Threaded tasks (I.e., MKL kernels)
  - MPI applications -> tasks that involve several nodes
  - Integration with BSC OmpSs
  - Streaming tasks for data flow executions
- Support to Failure Management
- Parallel Machine Learning with dislib
- Available in MareNostrum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



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

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



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Some interesting features

- Task constraints: enable to define HW or SW requirements

```
@constraint (MemorySize=6.0,
ProcessorPerformance="5000")
@task (c=INOUT)
def myfunc(a, b, c):
    ...
```

- Linking with other programming models:

```
@constraint (computingUnits= "248")
@mpi (runner="mpirun", computingNodes= "16", ...)
@task (returns=int, stdOutFile=FILE_OUT_STDOUT,
...) def nems(stdOutFile, stdErrFile):
    pass
```

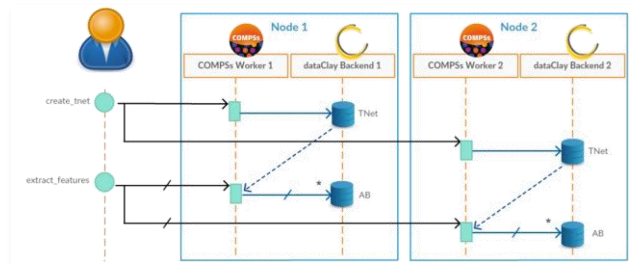
- Task failure management

```
@task(file_path=FILE_INOUT,
on_failure='CANCEL_SUCCESSORS')
def task(file_path):
    ...
    if cond :
        raise Exception()
```

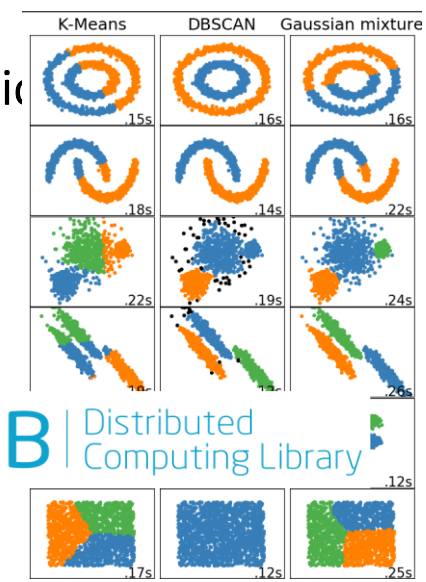
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Integration with Machine Learning

- Thanks to the Python interface, the integration with ML packages is smooth:
  - Tensorflow, PyTorch, ...
  - Tiramisu: transfer learning framework Tensorflow + PyCOMPSs + dataClay



- dislib: Collection of machine learning algorithms developed on top of PyCOMPSs
  - Unified interface, inspired in scikit-learn (fit-predic
  - Unified data acquisition methods and using an independent distributed data representation
  - Parallelism transparent to the user – PyCOMPSs parallelism hidden
  - Open source, available to the comm



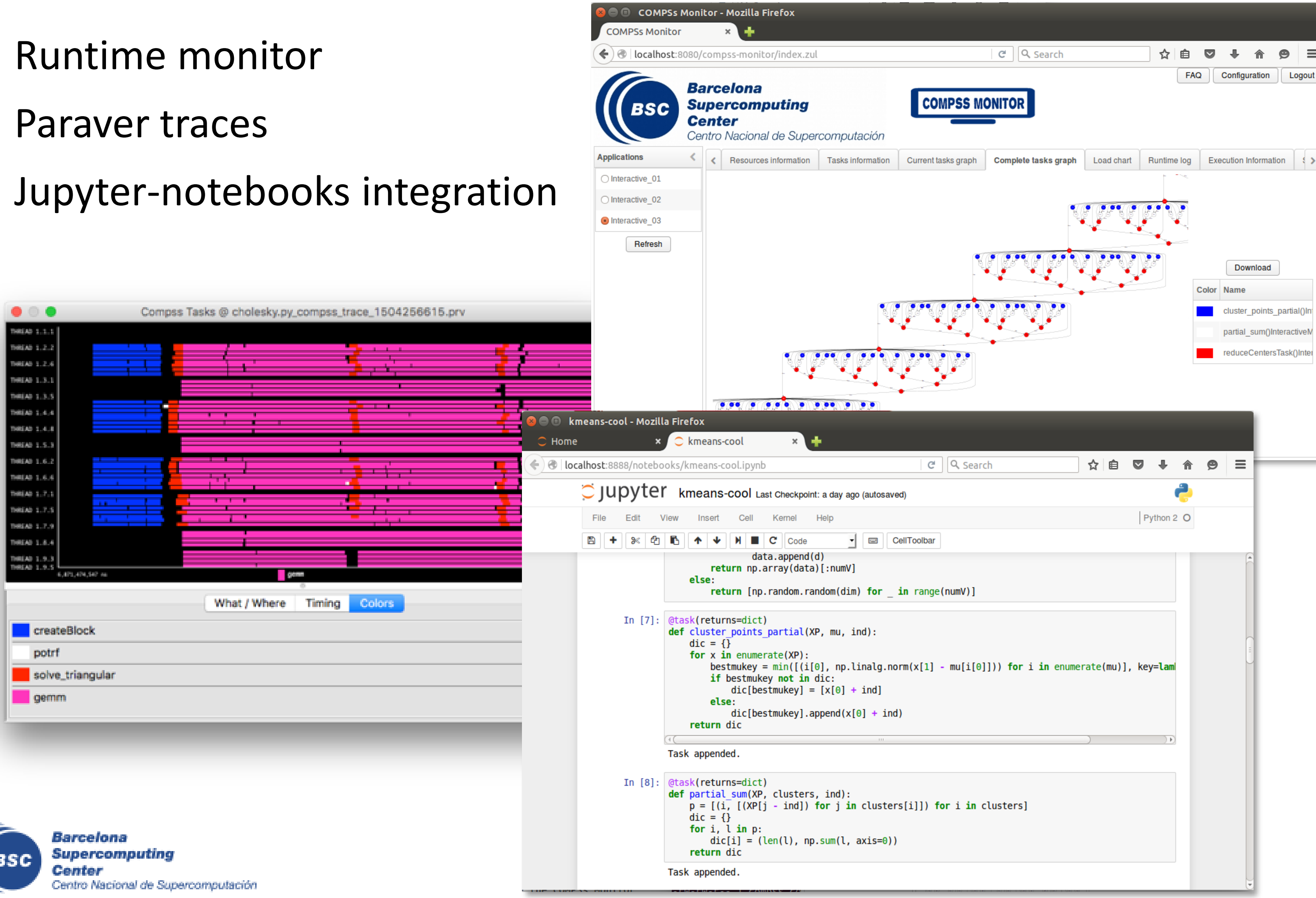
dislib.bsc.es

DISLIB Distributed Computing Library

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

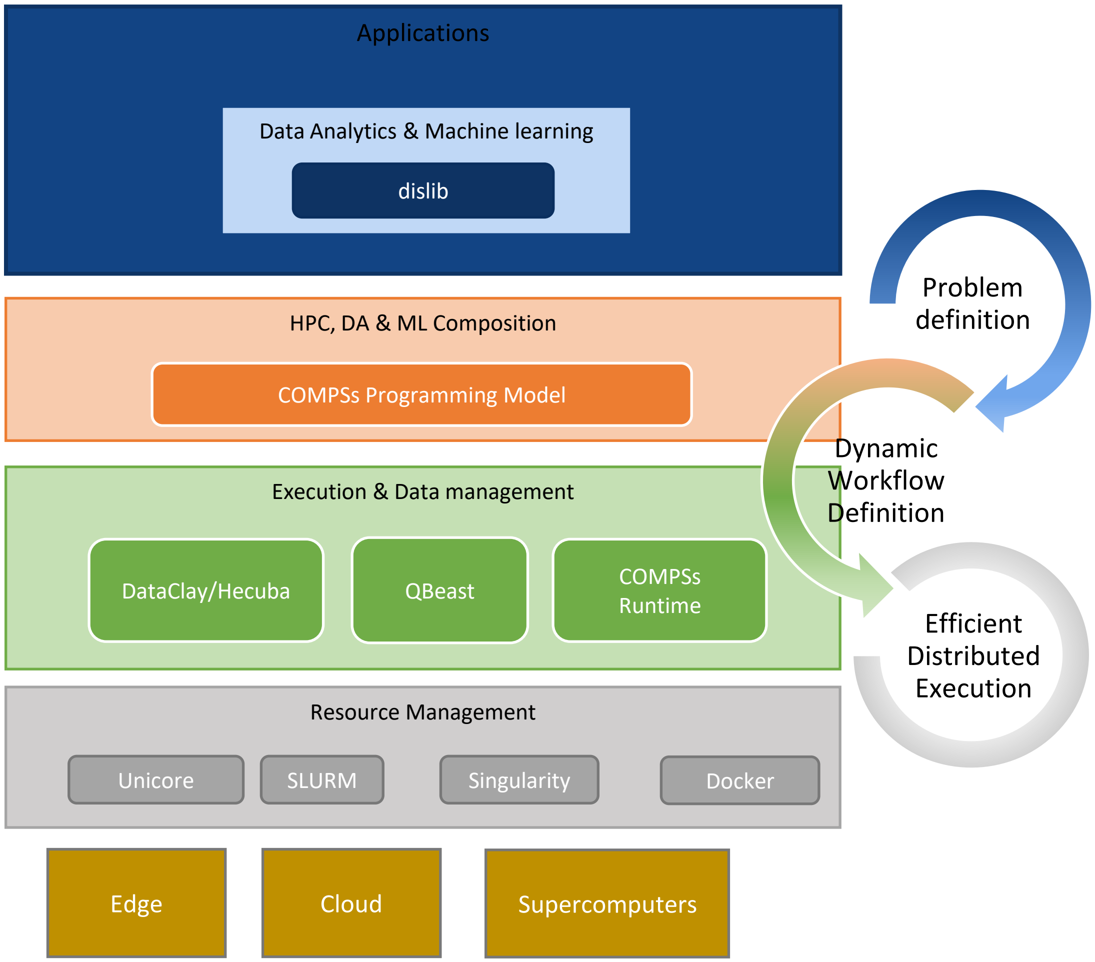
PyCOMPSs development environment

- 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.



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Projects where COMPSs is used/developed

AI SPRINT
AI IN SECURE PRIVACY-PRESERVING COMPUTING CONTINUUM

CAELESTIS

ICOS

HP2C-DT

COLMENA

DT-GEO

COMPSs

Per Med CoE
HPC/Exascale Centre of Excellence in Personalised Medicine

MEEP | MareNostrum Experimental Exascale Platform

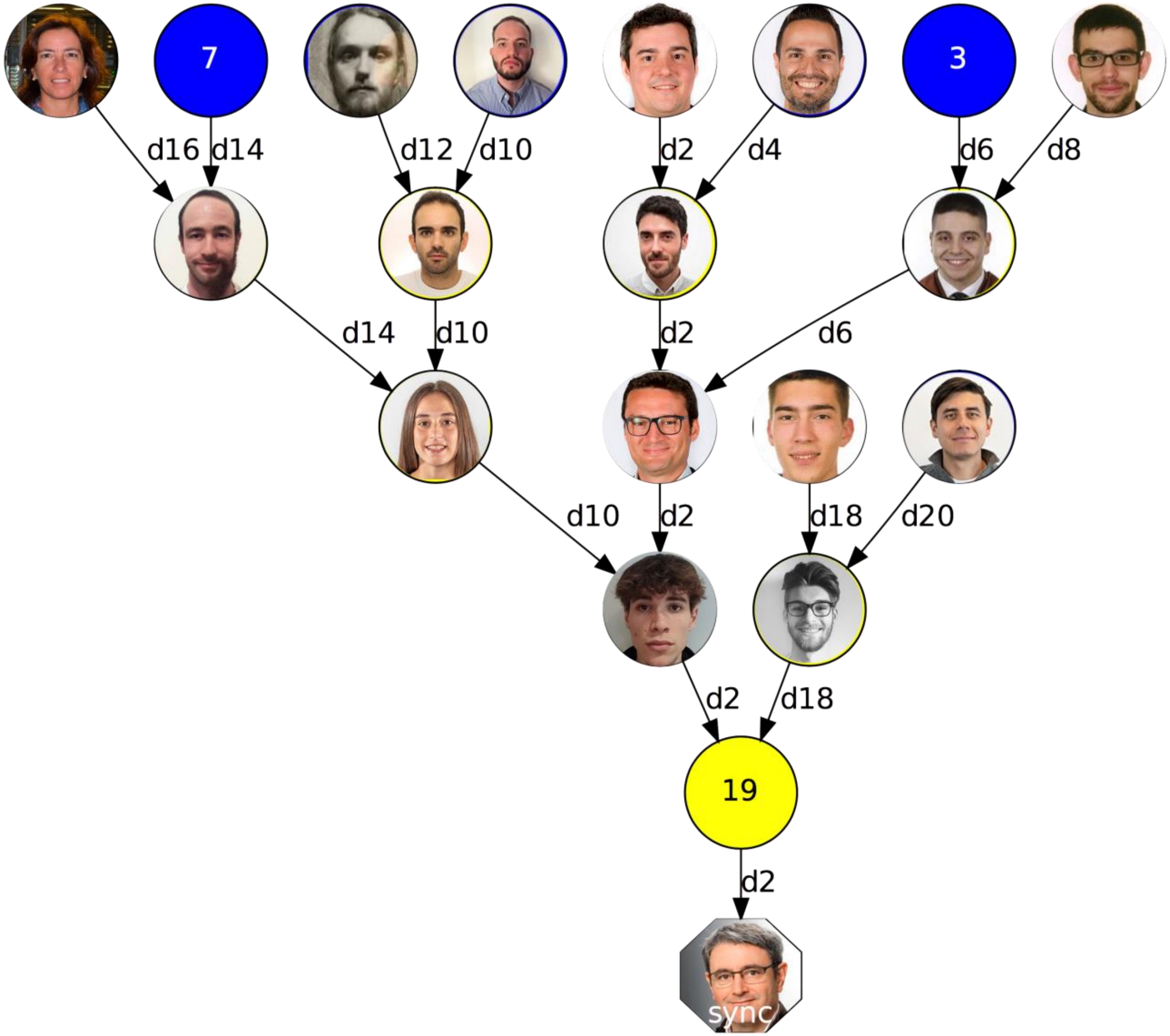
CEEC

eFlows4HPC

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Joint Laboratory for Extreme-Scale Computing
Inria NCSA Argonne BSC JÜLICH RIKEN

The WDC team



http://compss.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación