Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Programming parallel codes with PyCOMPSs

Rosa M Badia

28 May 2020

Barcelona, Spain

www.bsc.es

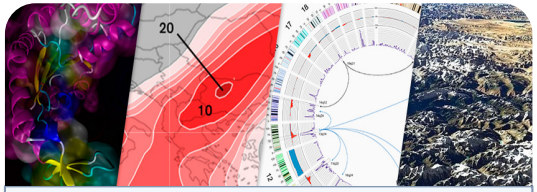
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Barcelona Supercomputing Center

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

BSC-CNS objectives

Supercomputing services to Spanish and EU researchers



R&D in Computer, Life, Earth and Engineering Sciences

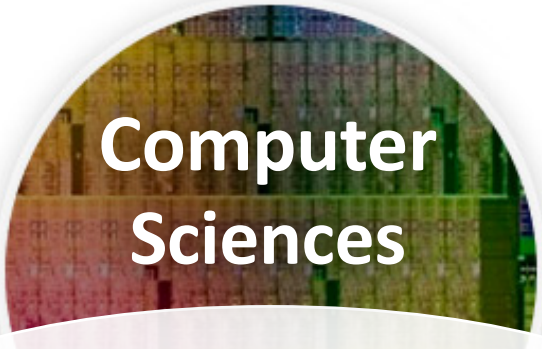
PhD programme, technology transfer, public engagement

BSC-CNS is a consortium that includes

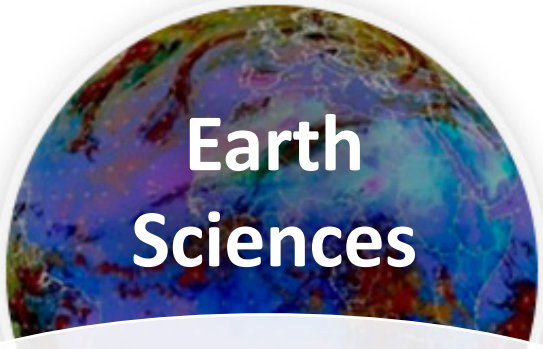
Spanish Government	60%	Gobierno de España · Ministerio de Economía y Competitividad
Catalonian Government	30%	Generalitat de Catalunya · Departament d'Empresa i Coneixement
Univ. Politècnica de Catalunya (UPC)	10%	Universitat Politècnica de Catalunya · BarcelonaTech

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Mission of BSC Scientific Departments



To influence the way machines are built, programmed and used: programming models, performance tools, Big Data, computer architecture, energy efficiency



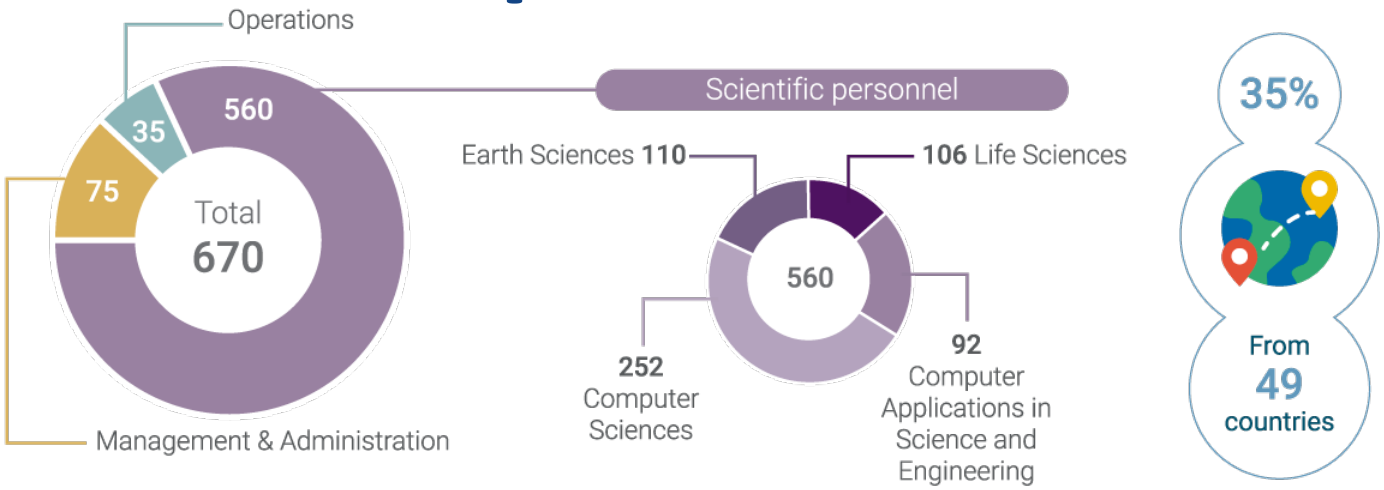
To develop and implement global and regional state-of-the-art models for short-term air quality forecast and long-term climate applications

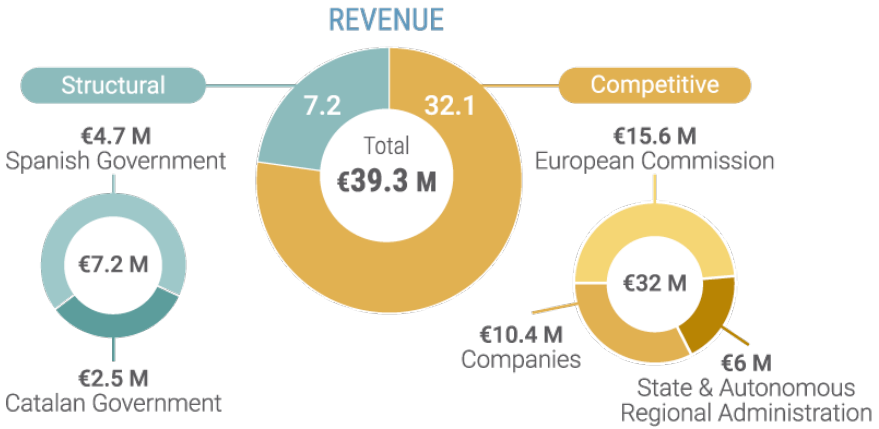
To understand living organisms by means of theoretical and computational methods (molecular modeling, genomics, proteomics)

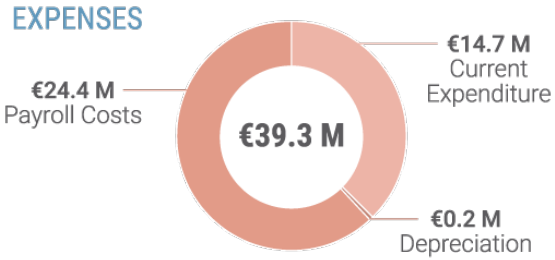
To develop scientific and engineering software to efficiently exploit super-computing capabilities (biomedical, geophysics, atmospheric, energy, social and economic simulations)

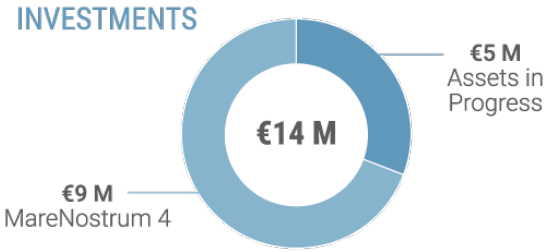
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

People and resources



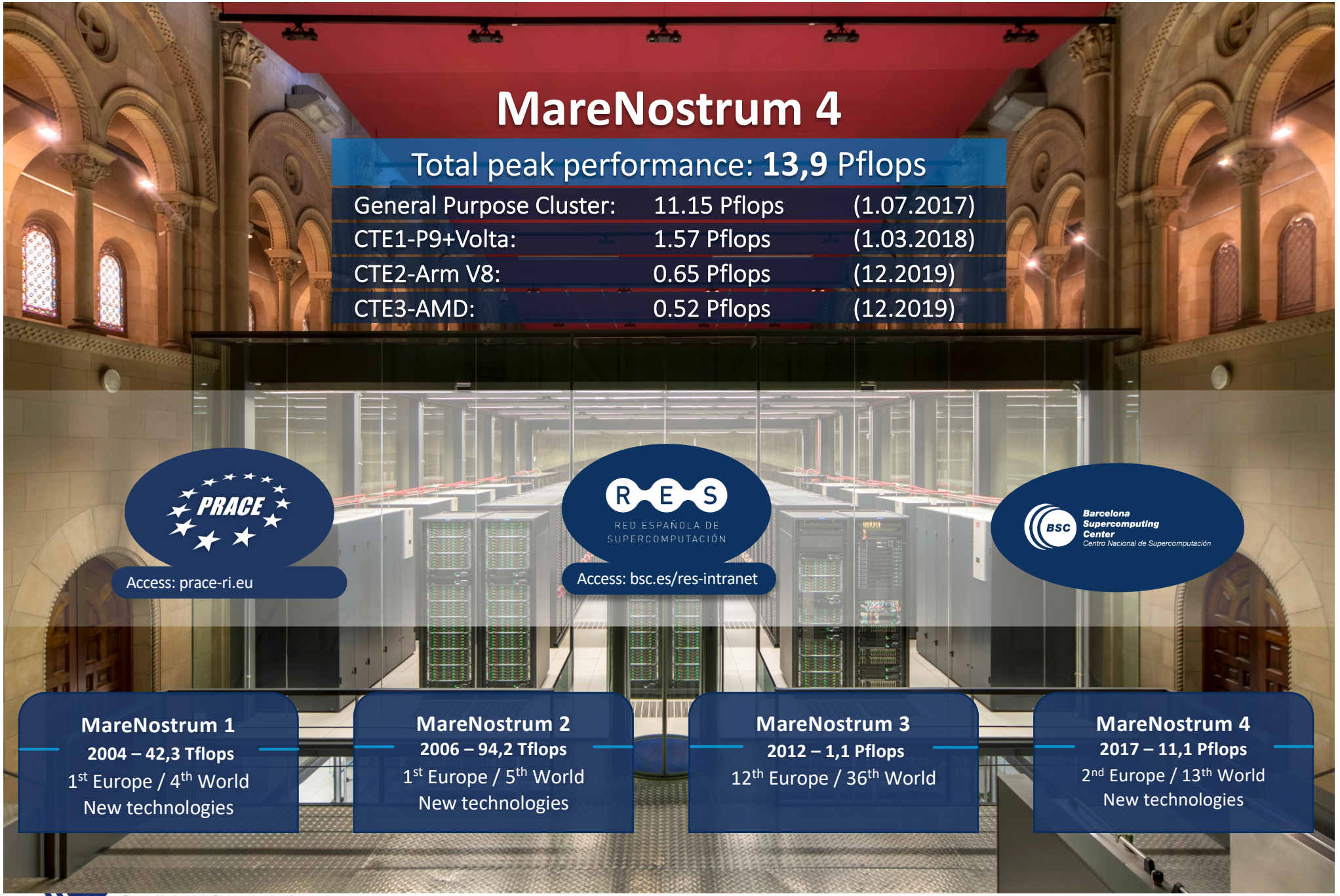






Barcelona Supercomputing Center
Centro Nacional de Supercomputación

December 31, 2019



Centro Nacional de Supercomputación

General purpose cluster

Rpeak: 11.15 Pflops
384.75 TB of main memory
3,456 nodes:
- 2x Intel Xeon Platinum 8160
- 216 nodes with 12x32 GB DDR4
- 3240 nodes with 12x8 GB DDR4

Interconnection networks:
- 100Gb Intel Omni-Path Full-Fat Tree
- 10Gb Ethernet

Operating System: SUSE Linux Enterprise Server 12 SP2

MN4 CTE1-P9+Volta

Rpeak: 1.57 Pflops
2 login node and 52 compute nodes:
- 2 x IBM Power9, 20 cores and 4 threads/core, total 160
- 4 x GPU NVIDIA V100 (Volta)
- 512GB of main memory
- 2 x SSD 1.9TB as local storage
- 2 x 3.2TB NVME

GPFS via one fiber link 10 GBit Infiniband interconnection network
The operating system is Red Hat Enterprise Linux Server 7.5 (Maipo).

MN4 CTE AMD (Lenovo)

Rpeak: 0.51 Pflops
34 nodes in 2 racks
- 1xAMD Rome, 64c (estimated 1750 Gflops)
- 2xAMD Radeon Instinct MI500
  - 32GB HMB2 memory each with 6.7 TFlops
- 1024 GB
- 480 GB SSD

34 TB RAM
30 KW*
Mellanox Infiniband HDR100, full non-blocking
DLC-Direct Liquid Cooling

MN4 CTE ARM (Fujitsu)

Rpeak: 0.65 Pflops
Post-FX100
192 nodes in 1/2 rack
- 1xArmv8 + SVE
- 32 GB

6 TB RAM
?? KW*
Tofu Inteconnect & IB EDR
DLC-Direct Liquid Cooling

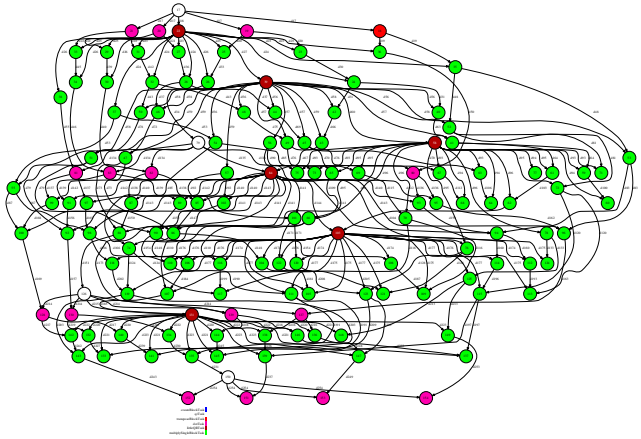
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

www.bsc.es

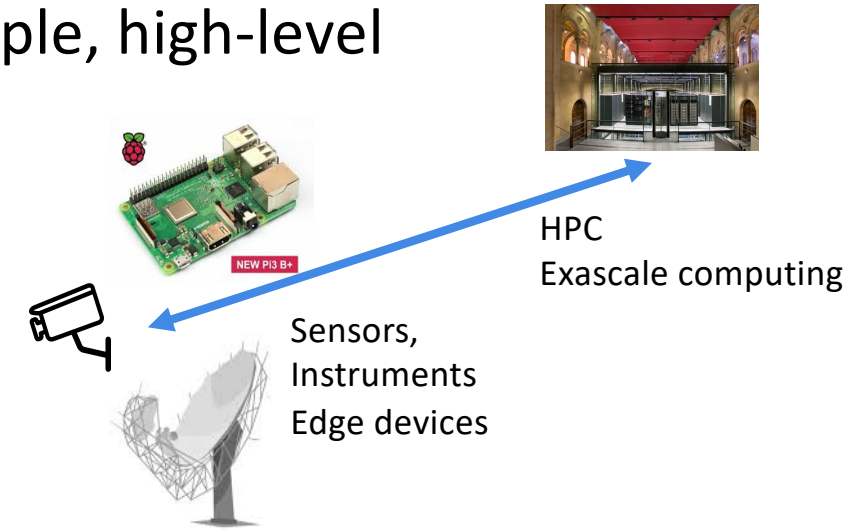
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Programming with PyCOMSs/COMPSs

PyCOMPSs/COMPSs ambition



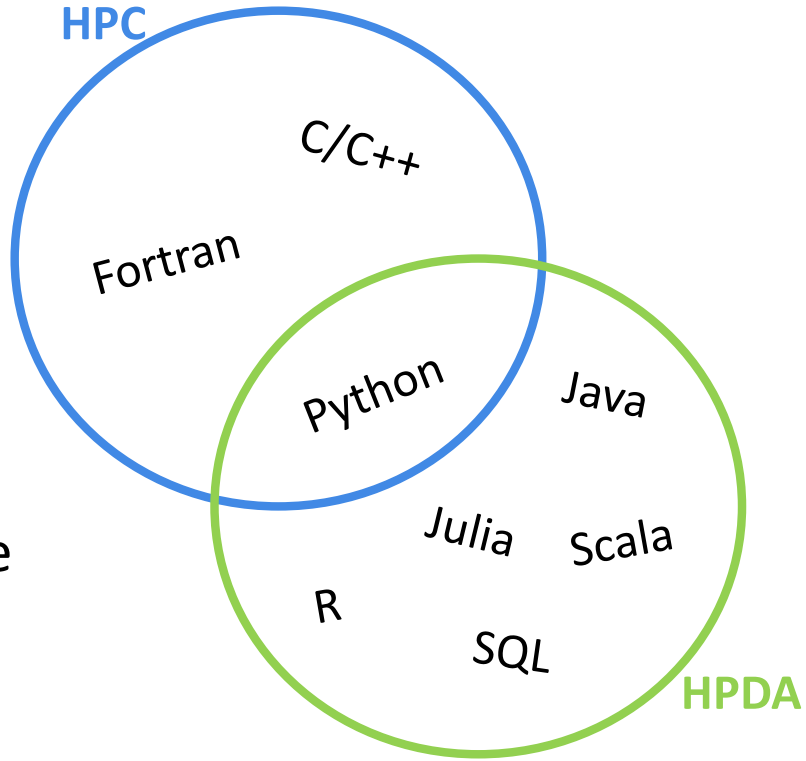
- Complex infrastructures with multiple and heterogeneous components
- Complex applications, composed of multiple components and pieces of software
- How to describe the workflows in such environment?
- Holistic approach where both data and computing are integrated in a single flow built on simple, high-level interfaces
  - Integration of computational workloads, with machine learning and data analytics
  - Intelligent runtime that can make scheduling and allocation, data-transfer, and other decisions



Why Python?

Python is powerful... and fast;
plays well with others;
runs everywhere;
is friendly & easy to learn;
*is Open.**

- Emphasizes code readability, its syntax allows programmers to express concepts in fewer lines of code
- Large community using it, including scientific and numeric
- Large number of software modules available
- Very well integrated with data analytics and machine learning (Tensorflow, PyTorch, dask, scikit-learn, ...)
- Intersection with HPC and data analytics programming languages

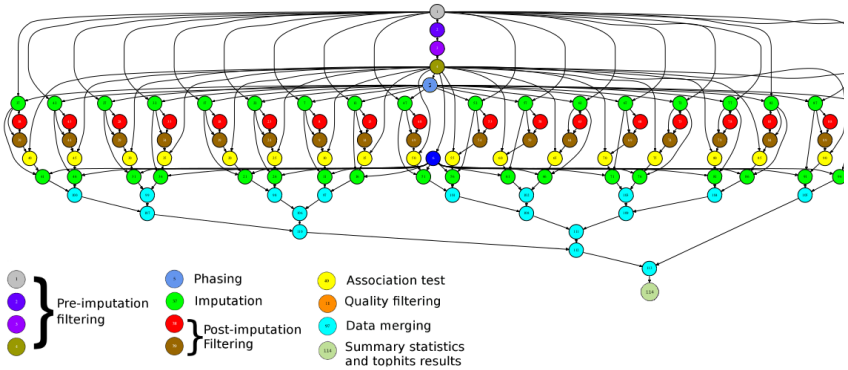


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

* From python.org

Programming with PyCOMPSs/COMPSs

- Sequential programming, parallel execution
- General purpose programming language + annotations/hints
  - To identify tasks and directionality of data
  - Task based: task is the unit of work
- Builds a task graph at runtime that express potential concurrency
- Exploitation of parallelism
  - … and of parallelism created later on
- Simple linear address space
- Agnostic of computing platform
  - Runtime takes all scheduling and data transfer decisions



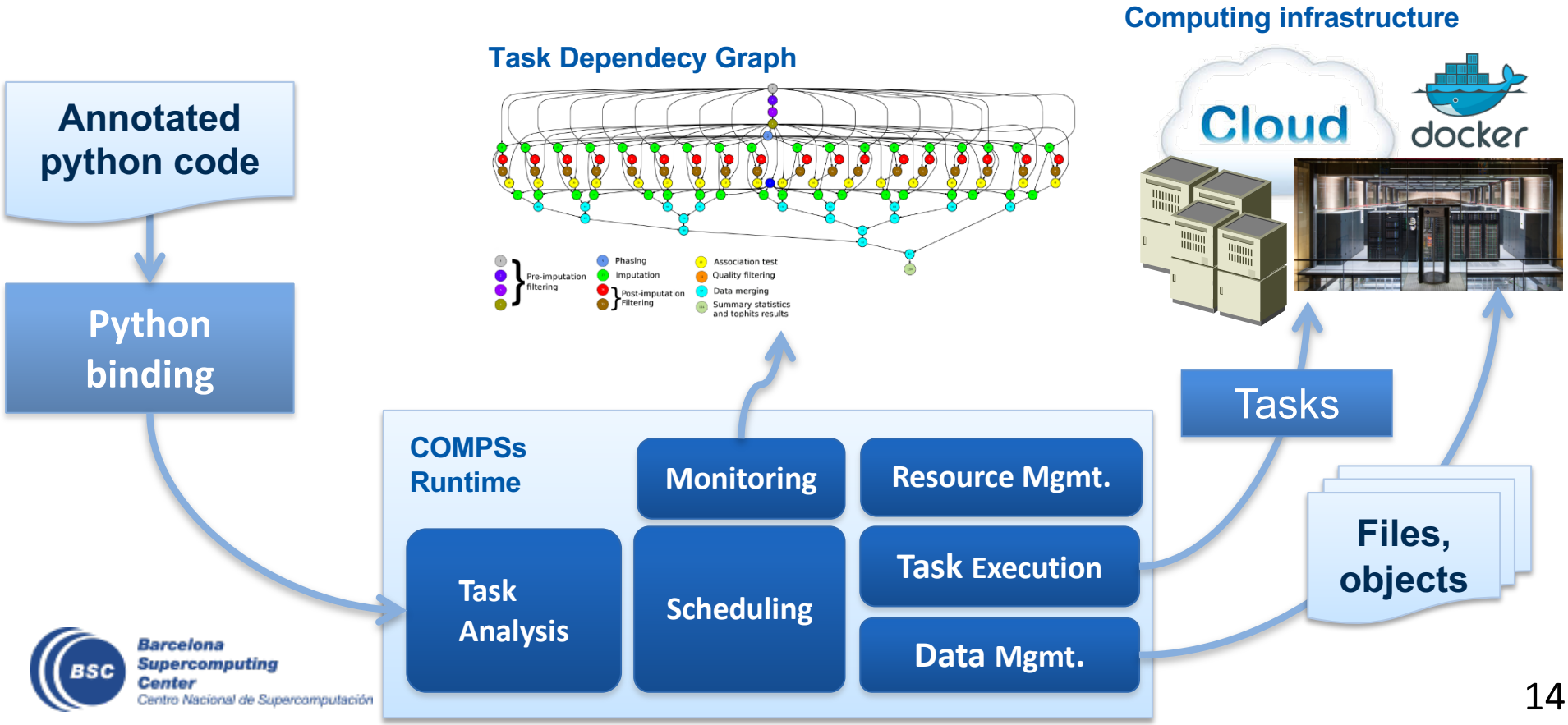

```
@task(c=INOUT)
def multiply(a, b, c):
    c += a*b
```

```
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][
compss_barrier()
mulTime = time.time() - startMulTime
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

PyCOMPSs/COMPSs runtime

- PyCOMPSs/COMPSs applications executed in distributed mode following the master-worker paradigm
  - Description of computational infrastructure in an XML file
- Sequential execution starts in master node and tasks are offloaded to worker nodes
- All data scheduling decisions and data transfers are performed by the runtime



Other decorators: Tasks' constraints and versions

- Constraints enable to define HW or SW features required to execute a task
  - Runtime performs the match-making between the task and the computing nodes
  - Support for multi-core tasks and for tasks with memory constraints
  - **Support for the heterogeneity of the platform**
- Versions: Mechanism to support multiple implementations of a given behavior (polymorphism)
  - **Runtime selects most appropriate device in the platform to execute each task**

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

```
@implement (source class="myclass", method="myfunc")
@constraint (MemorySize=1.0, ProcessorType ="ARM")
@task (c=INOUT)
def myfunc_in_the_edge (a, b, c):
    ...
```

Example use of constraints: Guidance

- Guidance is a tool for Genome-Wide Association Studies (GWAS) (Develop by BSC, with other collaborators)
- This workflow composes multiple external binaries and scripts
- Different tasks, although sequential, have different high memory requirements


```
@constraint (MemorySize=6.0)
@task (gmapFile=FILE_IN, knownHapFile=FILE_IN, ...)
def imputeWithImpute(gmapFile, knownHapFile, theChromo, ...):
...
```

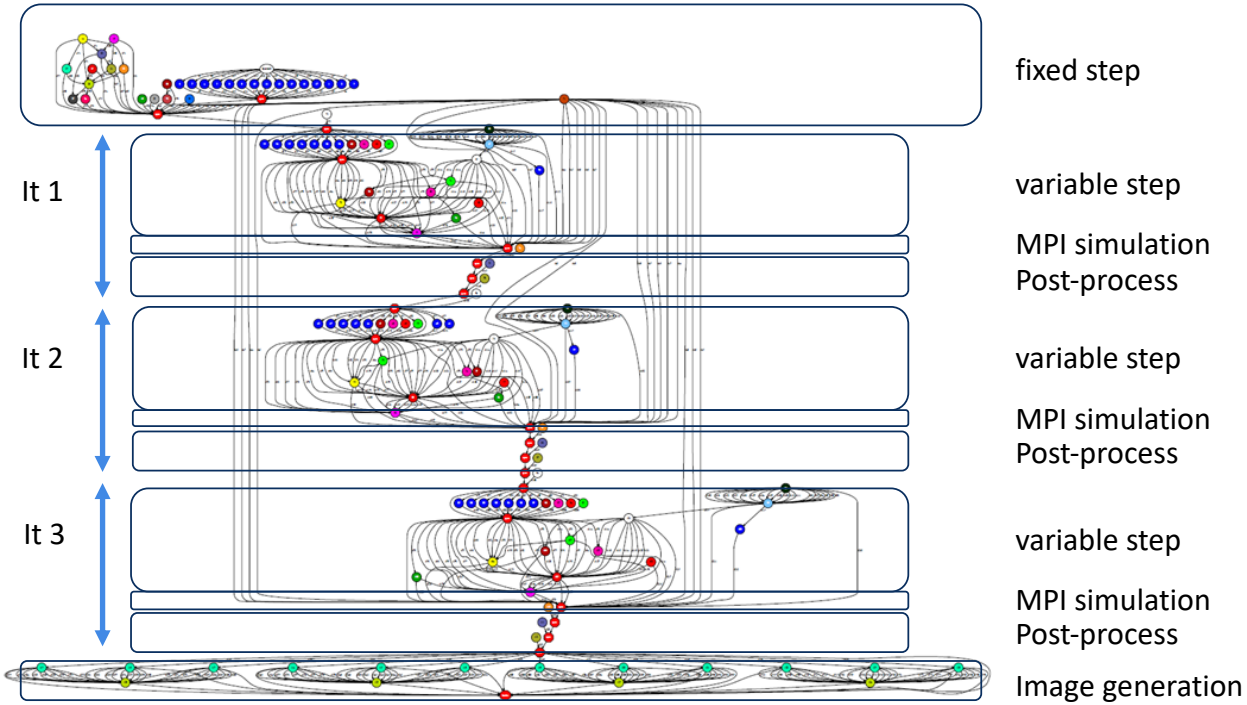
Other decorators: linking with other programming models

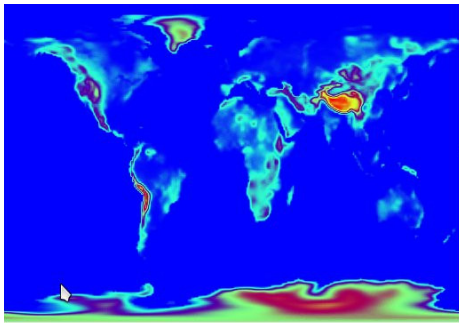
- A task can be more than a sequential function
  - A task in PyCOMPSs can be **sequential, multicore or multi-node**
  - External binary invocation: wrapper function generated automatically
  - Supports for alternative programming models: MPI and OmpSs
- Additional decorators:
  - @binary(binary="app.bin")
  - @ompss(binary="ompssApp.bin")
  - @mpi(binary="mpiApp.bin", runner="mpirun", computingNodes=8)
- Can be combined with the @constraint and @implement decorators

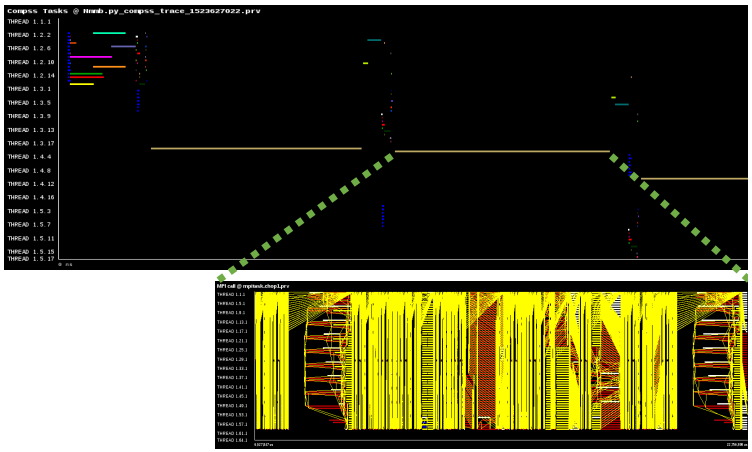
```
@constraint (computingUnits= "248")
@mpi (runner="mpirun", computingNodes= "16", ...)
@task (returns=int, stdOutFile=FILE_OUT_STDOUT, ...)
def nems(stdOutFile, stdErrFile):
    pass
```

NMMB-Monarch: Weather and dust forecast

- Multiscale Online Nonhydrostatic Atmosphere Chemistry model
  - Weather and dust forecast
  - Multiscale: Global to regional (up to 1km) scales allowed
  - Combines multiple scripts with the invocation of an MPI weather simulator







Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Failure management and exceptions

- New interface than enables the programmer to give hints about failure management or define timeouts to tasks
  - Options: RETRY, CANCEL_SUCCESSORS, FAIL, IGNORE

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

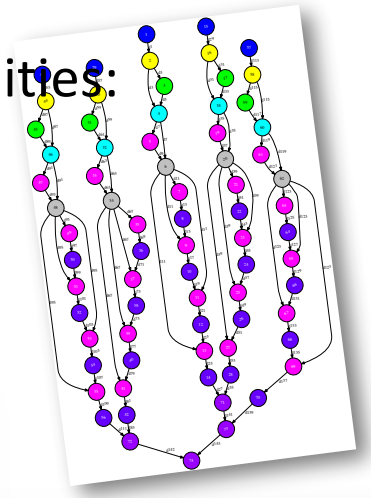
- Task exceptions: enables to cancel a group of tasks on the occurrence of an exception

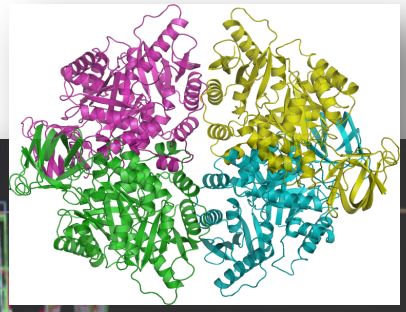
```
@task(file_path=FILE_INOUT)
def comp_task(file_path):
    ...
    raise COMPSsException("Exception")
```

```
def test_cancellation(file_name):
    try:
        with TaskGroup('failedGroup'):
            long_task(file_name)
            long_task(file_name)
            executed_task(file_name)
            comp_task(file_name)
    except COMPSsException:
        print("COMPSsException caught")
        write_two(file_name)
    write_two(file_name)
```

BioExcel Building Blocks (biobb)

- BioBB: Python building blocks that create new layer of compatibility and interoperability over popular bioinformatics tools
- Current version version of BioBB uses PyCOMPSs functionalities:
  - Execution of multi-node tasks
  - Task failure management
  - Task timeouts
- Massive parallel execution with bibobb and PyCOMPSs
  - Pyruvate Kinase protein
  - Simulation of 200 variants
  - ~395.000 atoms MD
  - GROMACS MD runs executed as 4 nodes MPI simulations (192 processes each)
  - Executed with PyCOMPSs, with 800 nodes of MareNostrum 4 (38,400 cores)



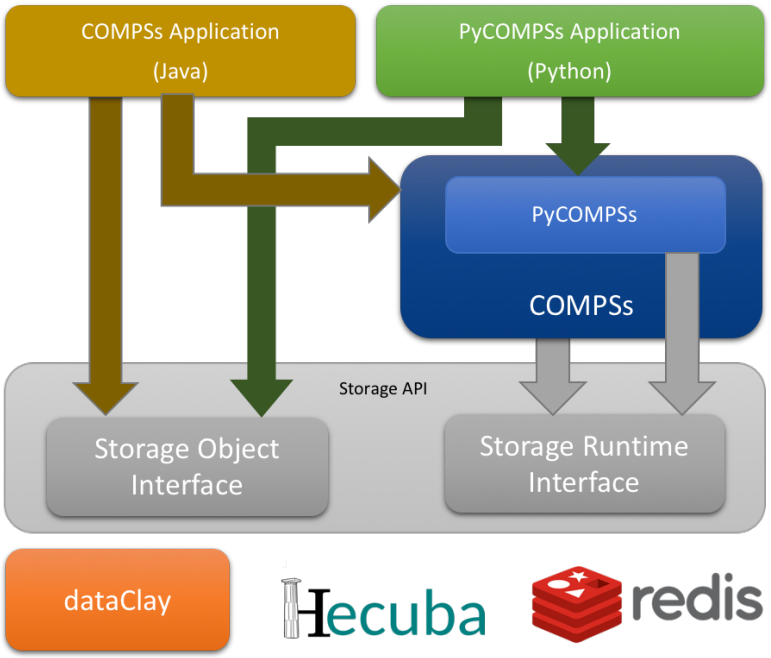


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Integration with persistent memory

- Persistent storage layer that offers a shared memory view on a distributed platform
  - Programmer may decide to make persistent specific objects in its code
  - Persistent objects are managed same way as regular objects
  - Tasks can operate with them

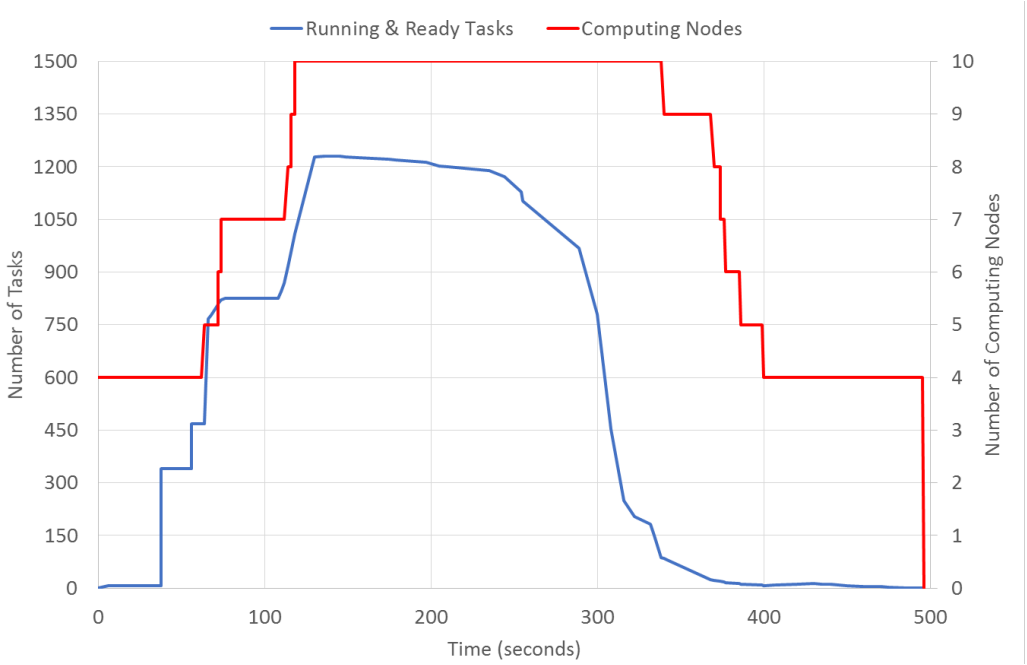
```
a = SampleClass ()
a.make_persistent()
Print a.func (3, 4)

a.mytask()
compss_barrier()

o = a.another_object
```

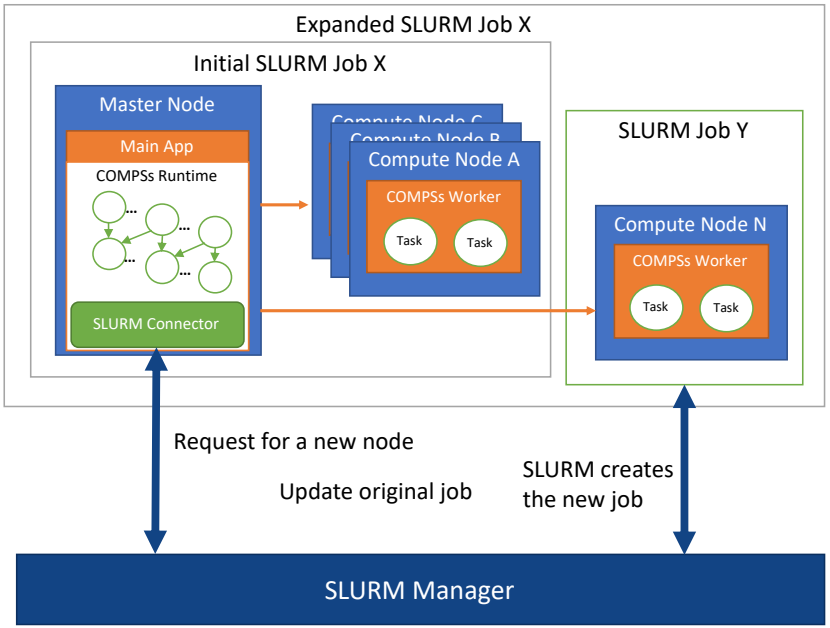


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Support for elasticity

- Possibility to adapt the computing infrastructure depending on the actual workload
- Now also for SLURM managed systems
- Feature that contributes to a more effective use of resources

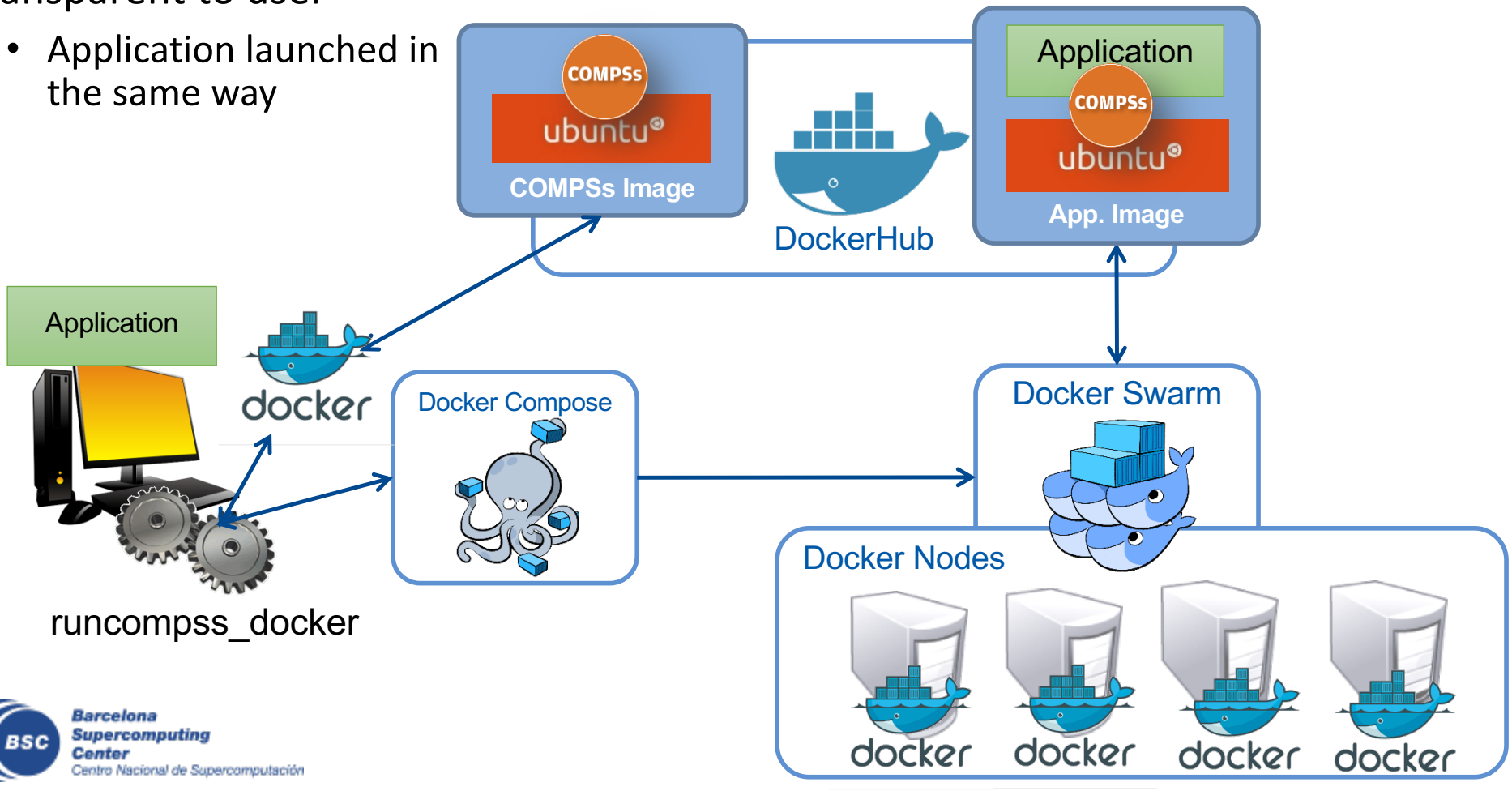




Barcelona Supercomputing Center
Centro Nacional de Supercomputación

COMPSs runtime: containers

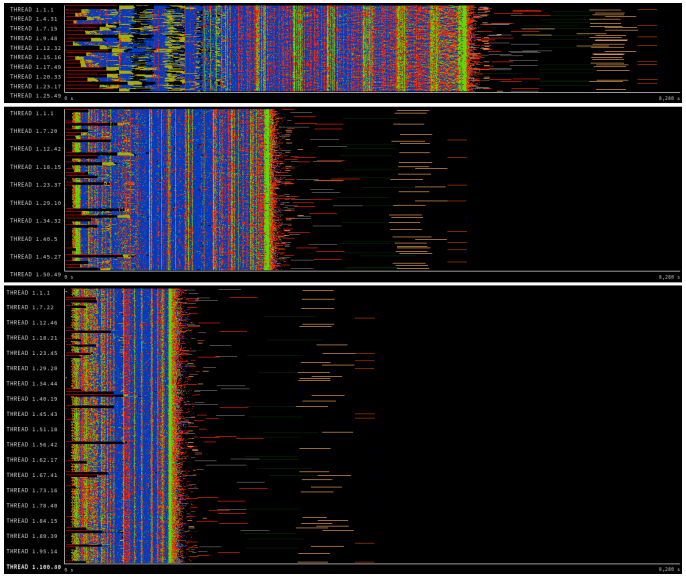
- Applications deployed as a set of docker container
  - Support for Singularity as well
- Support for elasticity
- Transparent to user
  - Application launched in the same way



COMPSs runtime: scalability and robustness

Guidance scalability

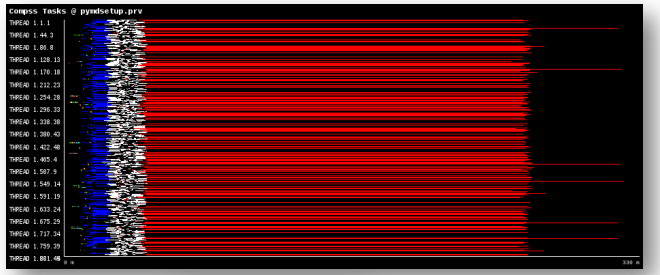
- 300 subjects, 2 genetic DBs, 8 diseases
- 170,152 tasks, around half a million files



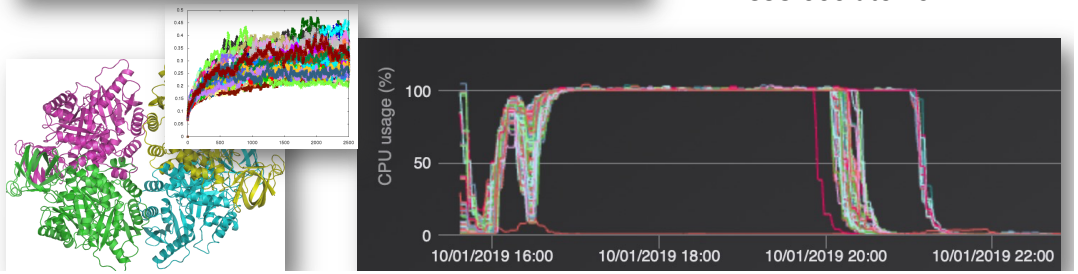
170,152 tasks, half a million files
Up to 100 nodes/4800 cores

BioExcel Building Blocks (biobb)

- Massively parallel execution with BioExcel building blocks and PyCOMPSs



PyMDsetup workflow
Pyruvate Kinase protein
Simulation of 200 variants
~395.000 atoms MD

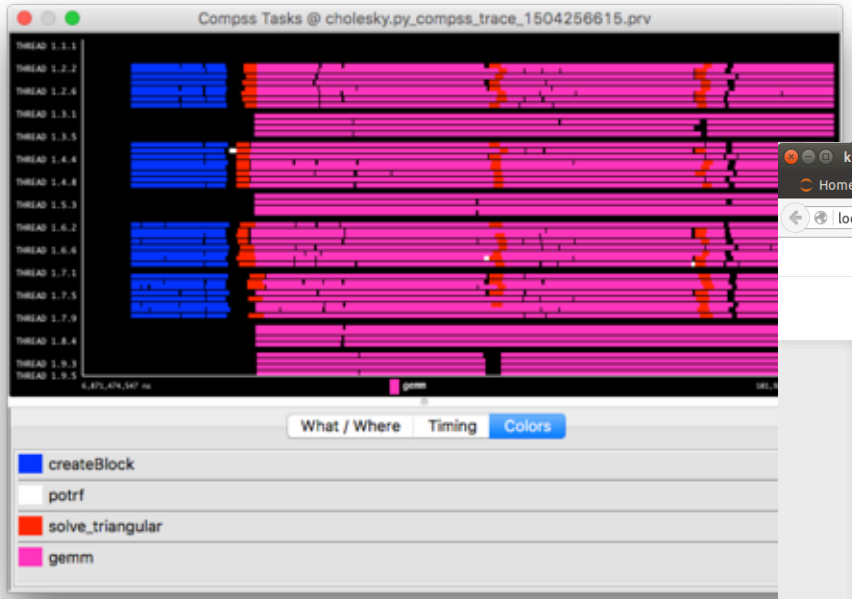


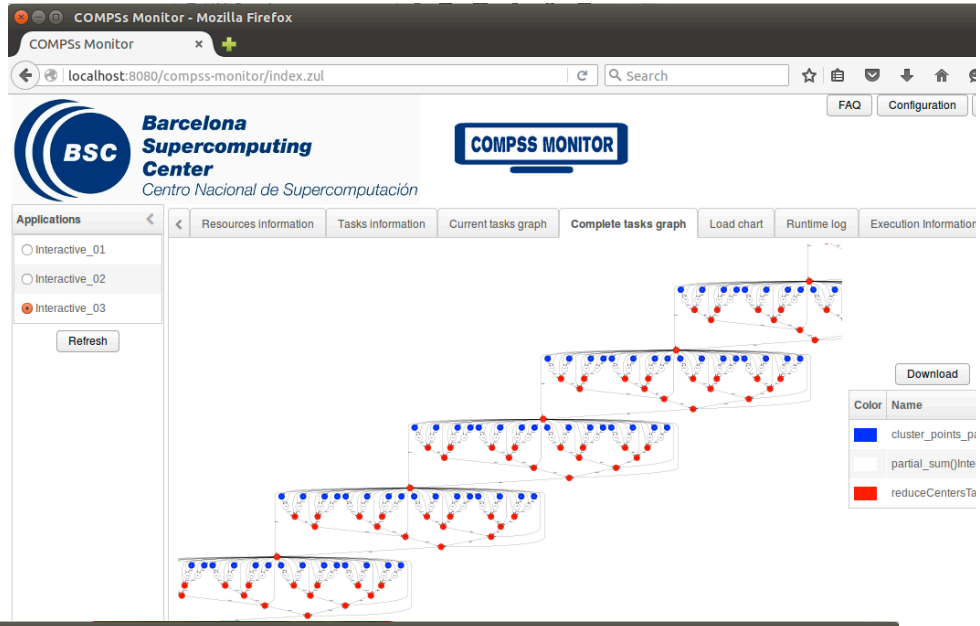
Workflow execution on 800 MN4 nodes (38400 cores)

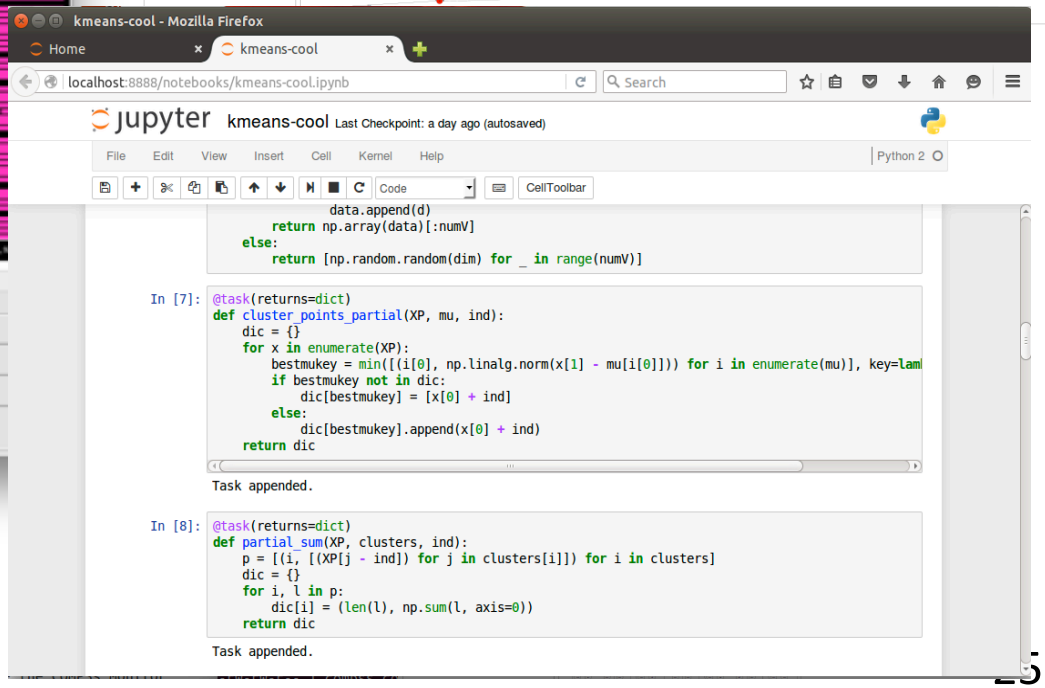
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

PyCOMPSs development environment

- Runtime monitor
- Paraver traces
- Jupyter-notebooks integration



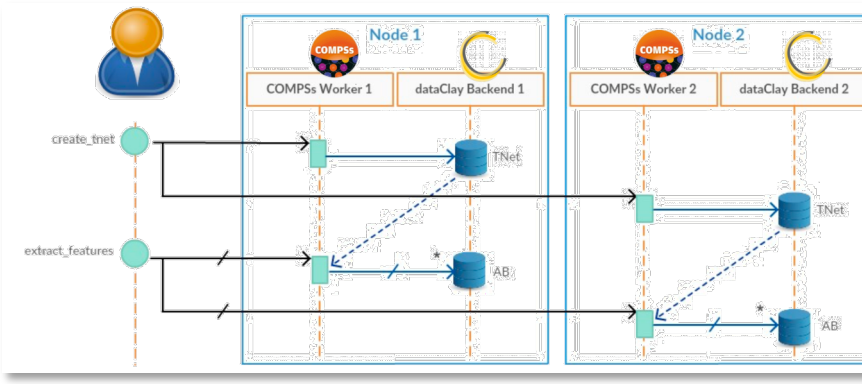




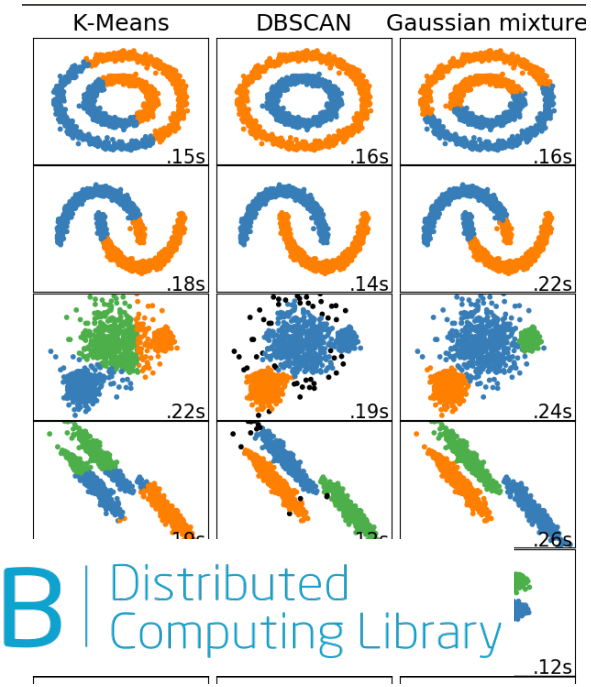
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

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-predict)
  - Unified data acquisition methods and using an independent distributed data representation
  - Parallelism transparent to the user – PyCOMPSs parallelism hidden
  - Open source, available to the community



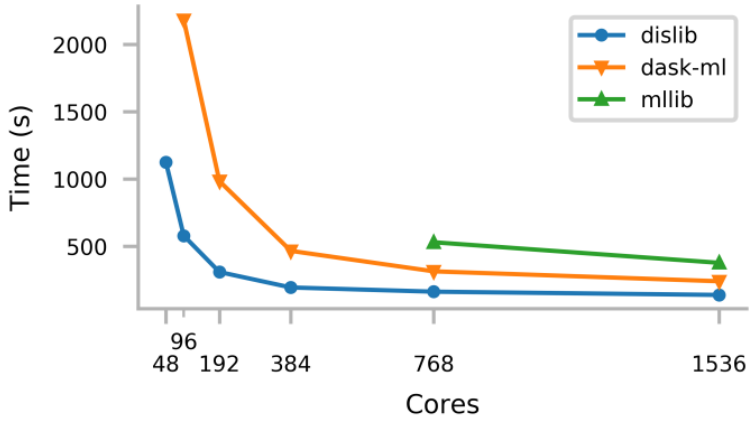
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

dislib.bsc.es

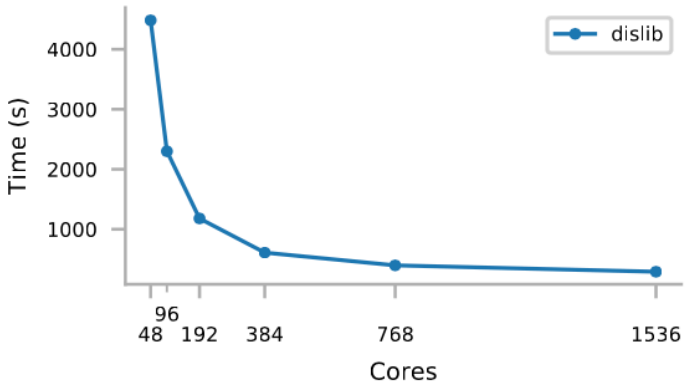
DISLIB | Distributed Computing Library

dislib sample results

- For very large sizes, dislib can obtain results while MLlib and dask fail to finish the execution

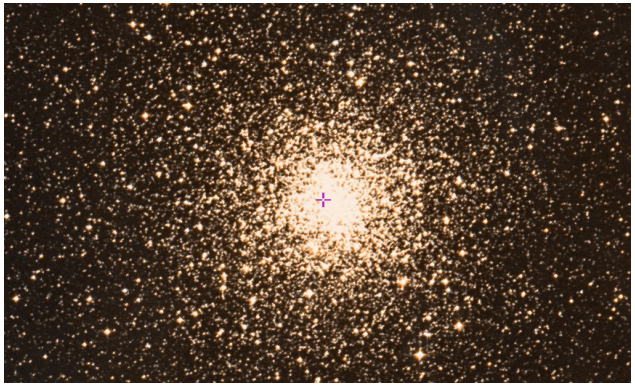


500 million samples
100 features



2 billion samples
100 features

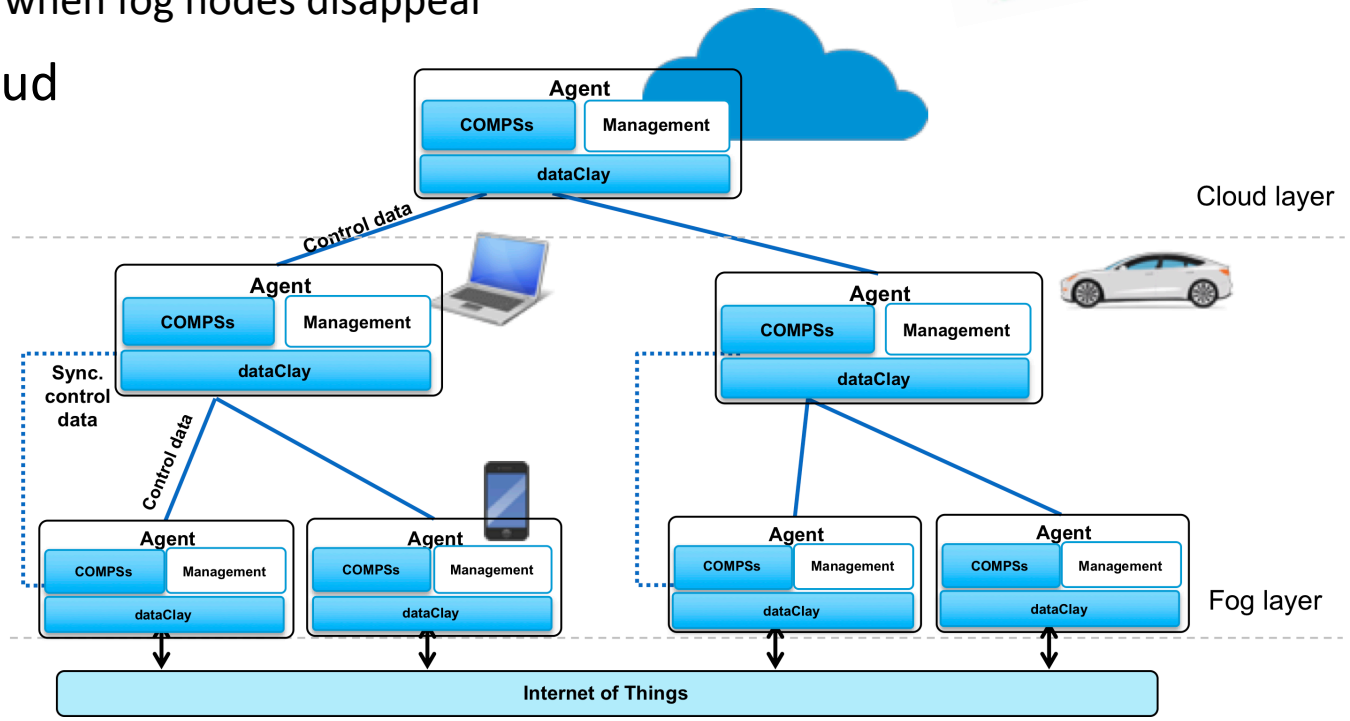
- Gaia satellite data: Sample scientific application:
  - Looking for open clusters in the sky with DBSCAN clustering
  - Subset of astrometric data from 2.5 million stars
    - Total data is 10^9 stars
  - Execution of 6,145 DBSCANs in parallel



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

COMPSs in a fog-to-cloud architecture

- **Decentralized** approach to deal with large amounts of data
- New COMPSs runtime handles distribution, parallelism and heterogeneity
- Runtime deployed as a microservice in an agent:
  - Agents are independent, can act as master or worker in an application execution, agents interact between them
  - Hierarchical structure
- Data managed by dataClay, in a federated mode
  - Support for data recovery when fog nodes disappear
- Fog-to-fog and Fog-to-cloud



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Conclusions

Summarizing

- PyCOMPSs/COMPSs is a task-based programming model that offers different user points of view
  - Fine grain Python tasks – linear algebra codes, machine learning algorithms (dislib)
  - Workflows of external binaries or composing large simulations as individual tasks (MPI)
  - Environment for the development of fog-to-cloud applications, based on a new distributed runtime
- The current PyCOMPSs/COMPSs roadmap focuses on its evolution to support the development of workflows in highly distributed computing platforms
- We aim to provide a programming environment able to integrate computational, analytics and machine learning codes

Further Information

- Project page: http://www.bsc.es/compss
  - Documentation
  - Virtual Appliance for testing & sample applications
  - Tutorials
- Source Code

  https://github.com/bsc-wdc/compss
- Docker Image

  https://hub.docker.com/r/compss/compss
- Applications

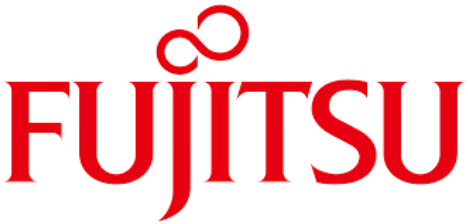
  https://github.com/bsc-wdc/apps

  https://github.com/bsc-wdc/dislib

Thanks!

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Projects where COMPSs is involved



www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Thanks!

BSC vision on programming models