Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Programming dynamic workflows in the Exascale Era

Daniele Lezzi

Workflows and Distributed Computing Group

ESiWACE2 Workshop on Emerging Technologies for Weather and Climate Modelling

30 June 2020

Challenges in highly distributed infrastructures

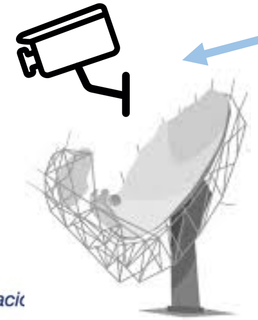
- Resources that appear and disappear
  - How to dynamically add/remove nodes to the infrastructure
- Heterogeneity
  - Different HW characteristics (performance, memory, etc)
  - Different architectures -> compilation issues
- Network
  - Different types of networks
  - Instability
- Trust and Security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

HPC
Exascale computing
Cloud

AI everywhere

Fog devices

Edge devices



Sensors
Instruments
Actuators

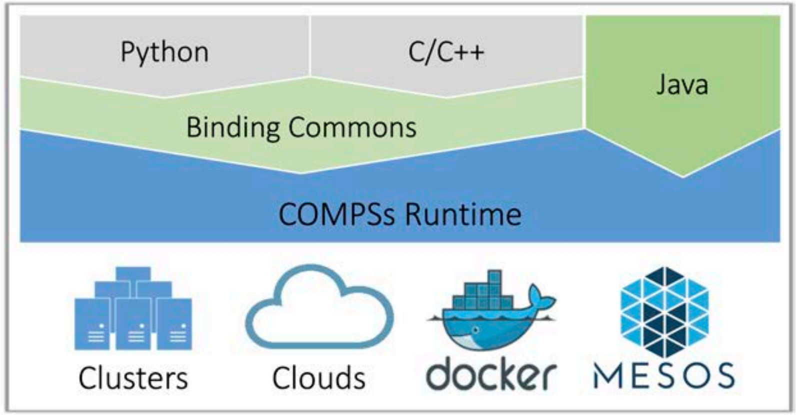
Barcelona Supercomputing Center
Centro Nacional de Supercomputació

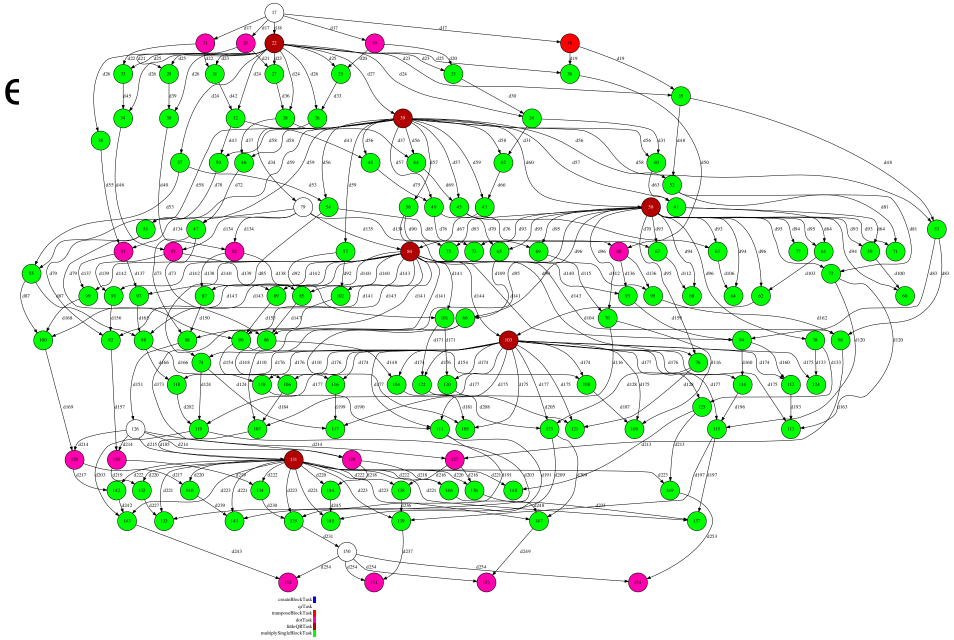
Data and storage challenge

- Sensors and instruments as sources of large amounts of heterogeneous data
  - Control of edge devices and remote access to sensor data
  - Edge devices typically have SDcards, much slower than SSD
- Compute and store close to the sensors
  - To avoid data transfers
  - For privacy/security aspects
- New data storage abstractions that enable access from the different devices
  - Object store versus file system?
  - Data reduction/lossy compression
- Task flow versus data flow:
  data streaming
- Metadata and traceability

Orchestration challenges

- How to describe the workflows in such environment? Which is the right interface?
- Focus:
  - Integration of computational workloads, with machine learning and data analytics
- Intelligent runtime that can make scheduling and allocation, data-transfer, and other decisions
- PyCOMPSs is a parallel task-based programming model for distributed computing platforms. Based on a sequential interface, at execution time the COMPSs runtime is able to exploit the inherent parallelism of applications at task level.

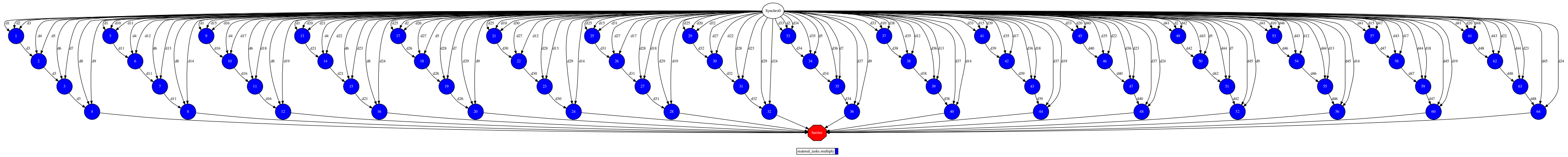




Barcelona Supercomputing Center
Centro Nacional de Supercomputación

PyCOMPSs Syntax

- Use of **decorators** to annotate tasks and to indicate arguments directionality
- Small API for data synchronization



Tasks definition

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

Main Program

```
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
```

Other decorators: Tasks’ constraints

- 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 heterogeneity on the devices in the platform**
- Versions: Mechanism to support multiple implementations of a given behavior (polymorphism)
  - **Runtime selects to execute the task in the most appropriate device in the platform**

```
@constraint (MemorySize=6.0, ProcessorPerformance=“5000”, ComputingUnits=“8”)
@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_other(a, b, c):
    ...
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Failure Management

- Interface than enables the programmer to give hints about failure management

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

- Options: RETRY, CANCEL_SUCCESSORS, FAIL, IGNORE
- Implications on file management:
  - I.e, on IGNORE, output files: are generated empty
- **Possibility of ignoring part of the execution of the workflow, for example if a task fails in an unstable device**
- **Opens the possibility of dynamic workflow behaviour depending on the actual outcome of the tasks**

Task Nesting

- Tasks that internally invoke new tasks

```
@compss(app_name="csvm-driver.py",
        computingNodes=”2”, …)
@constraint(ComputingUnits= “48”)
@task(returns=int)
def run_csvm(iflag="-i", i=1, cflag="-c", ...
    pass
```

- Why is task nesting relevant?
  - Support for structured programming
  - Reduces bottlenecks in scheduling and other runtime features
  - Better support for heterogeneous computing platforms

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Integration of HPC, ML and data analytics

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

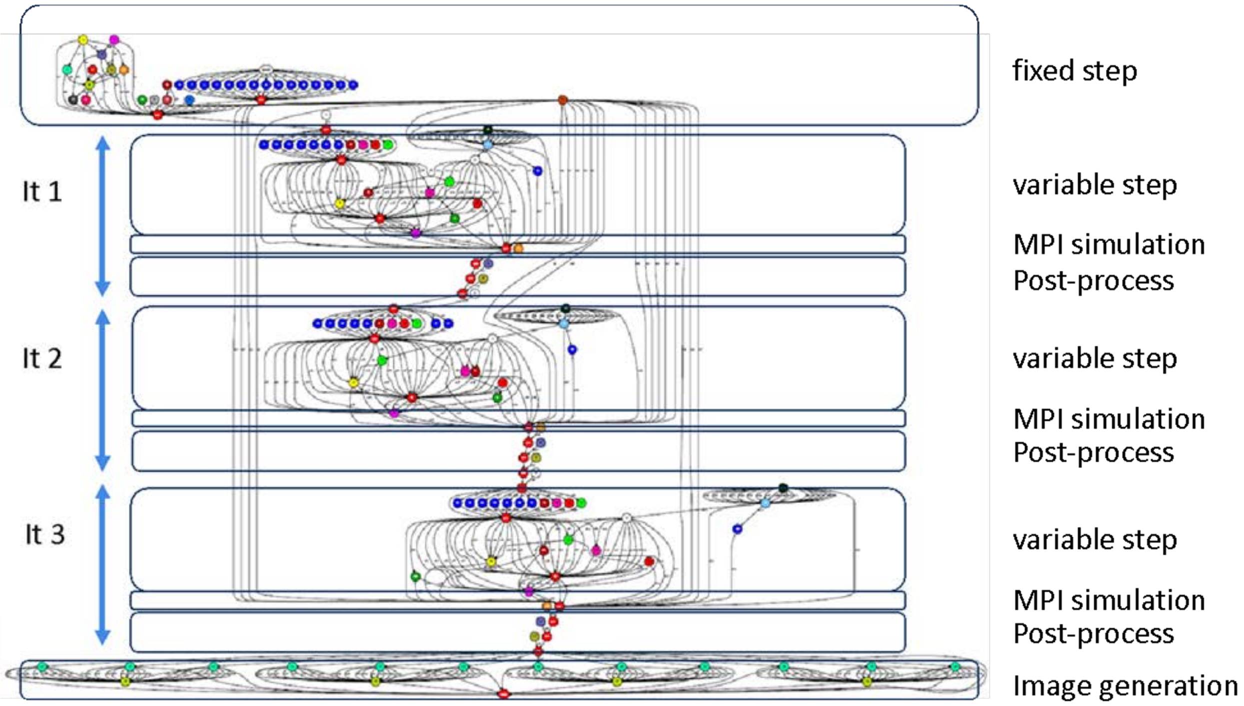
Other decorators: linking with other programming models

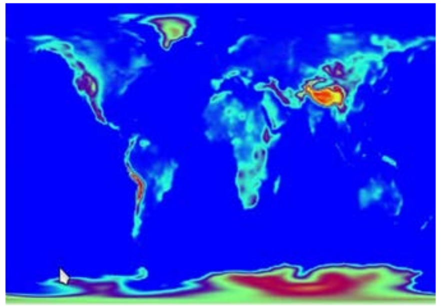
- A task can be more than a sequential function
  - A task in PyCOMPSs can be sequential, multicore or multi-node
  - External binary invocation: wrapper function generated automatically
  - Supports for alternative programming models: MPI and OmpSs
- Additional decorators:
  - @binary(binary="app.bin")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

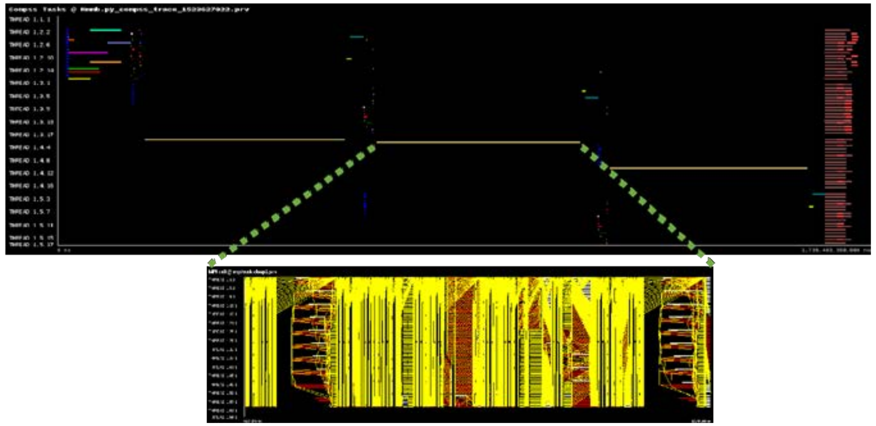
```
@constraint (computingUnits= "248")
@mpi (runner="mpirun", computingNodes= ”16”, ...)
@task (returns=int, stdOutFile=FILE_OUT_STDOUT, ...)
def nems(stdOutFile, stdErrFile):
    pass
```

NMMB-Monarch: Weather and dust forecast

- An example of usage of this idea is the application Multiscale Online Nonhydrostatic Atmosphere Chemistry model (NMMB-Monarch) aims at providing short to medium range weather and gas-phase chemistry forecasts from regional to global scales that performs weather and dust forecast. NMMB-Monarch is used as an operational tool to provide information services at BSC.
- The application combines multiple sequential scripts and MPI simulations. PyCOMPSs enables the smooth orchestration of all them as a single workflow.







Integration with persistent memory

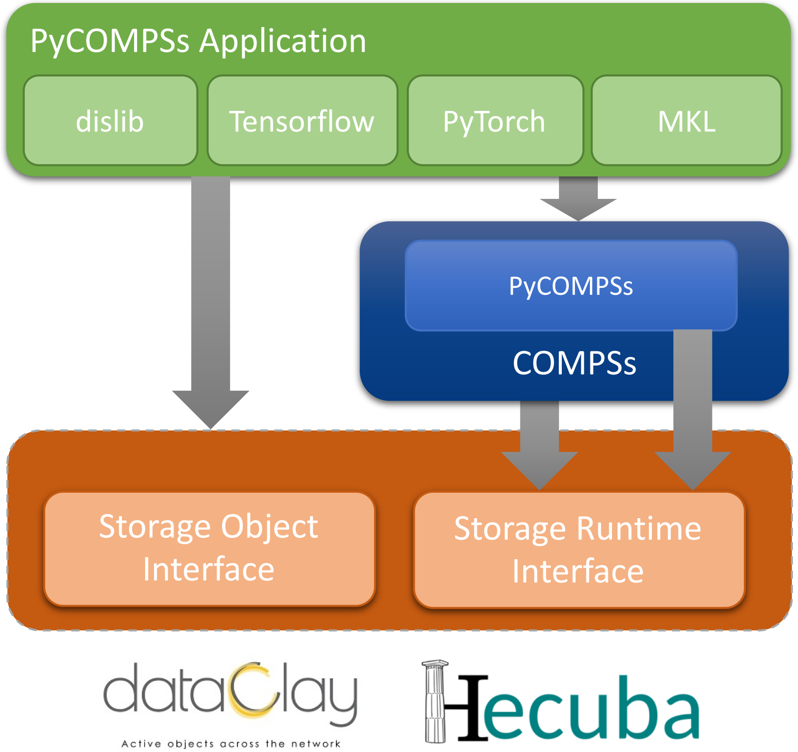
- Programmer may decide to make persistent specific objects in its code
- Persistent objects are managed same way as regular objects
- Tasks can operate with them

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

a.mytask()
compss_barrier()

o = a.another_object
```

- **Objects can be accessed/shared transparently in a distributed computing platform**

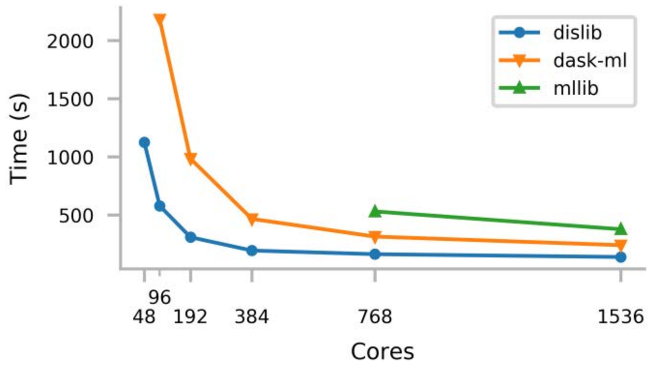


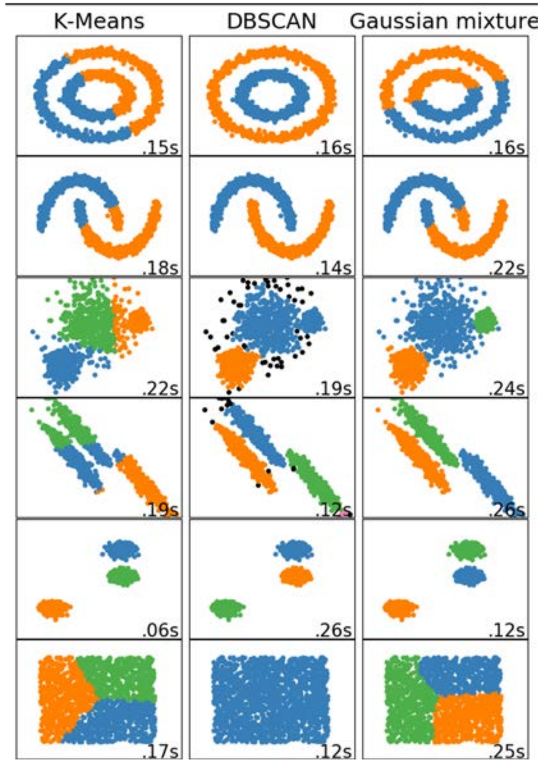
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Dislib: parallel machine learning

- Dislib, a distributed machine learning library parallelized with PyCOMPSs that enables large-scale data analytics on HPC infrastructures.
- Inspired by scikit-learn, dislib provides an estimator-based interface that improves productivity by making algorithms easy to use and interchangeable. This interface also makes programming with dislib very easy to scientists already familiar with scikit-learn.
- Dislib also provides a distributed data structure that can be operated as a regular Python object. The combination of this data structure and the estimator-based interface makes dislib a distributed version of scikit-learn, where communications, data transfers, and parallelism are automatically handled behind the scene

- 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





BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación

FUJITSU

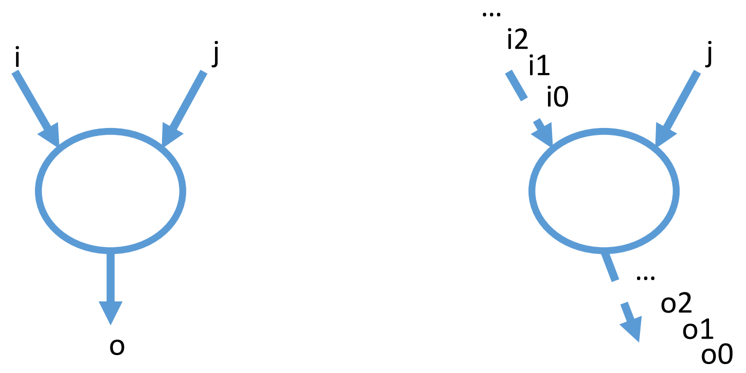
DISLIB Distributed Computing Library

Support for data streams: Task-flow versus data-flow

- New interface to support streaming data in tasks
- Task-flow and data-flow tasks live together in PyCOMPSs/COMPSs workflows
- Data-flow tasks persist while streams are not closed
  - Parameters can be one/multiple streams and non-streamed
- Runtime implementation based on Kafka

```
@task(fds=STREAM_OUT)
def sensor(fds):
        ...
    while not end():
        data = get_data_from_sensor()
        f.write(data)
    fds.close()
```

Task-flow task   Data-flow task



Task-flow task

- Receives data once
- Generates data once
- Does not persist in time
- Stateless tasks

Data-flow task

- Receives data multiple times
- Generates data multiple times
- Tasks persists in time
- Stateful tasks

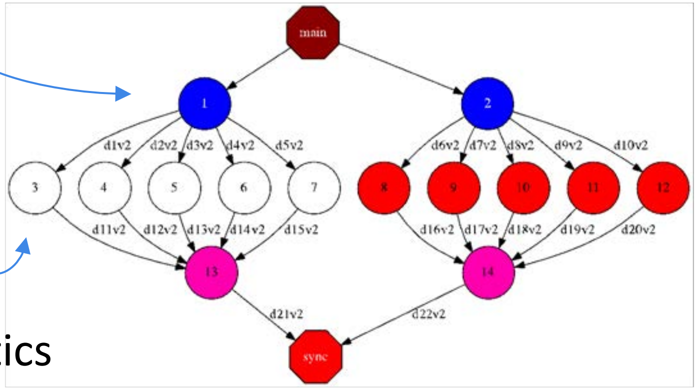
```
@task(fds_sensor=STREAM_IN, filtered=OUT)
def filter(fds_sensor, filtered):
        ...
    while not fds_sensor.is_closed():
        get_and_filter(fds_sensor, filtered)
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

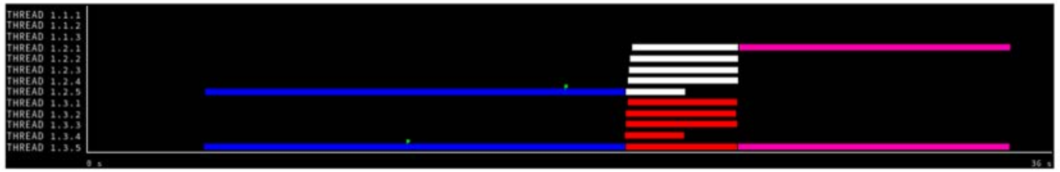
Use case: MPI simulations, analytics and streaming

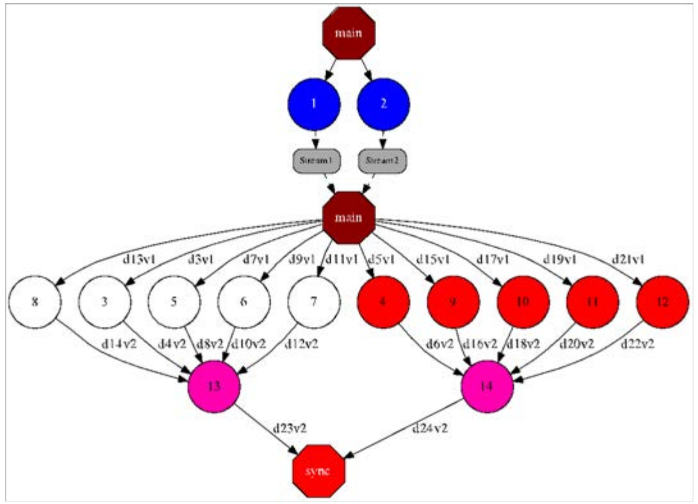
- Sample case: MPI simulations generating at given steps data to be processed by some analytics

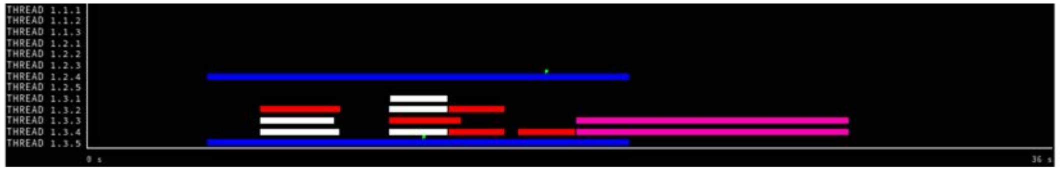
MPI



Analytics



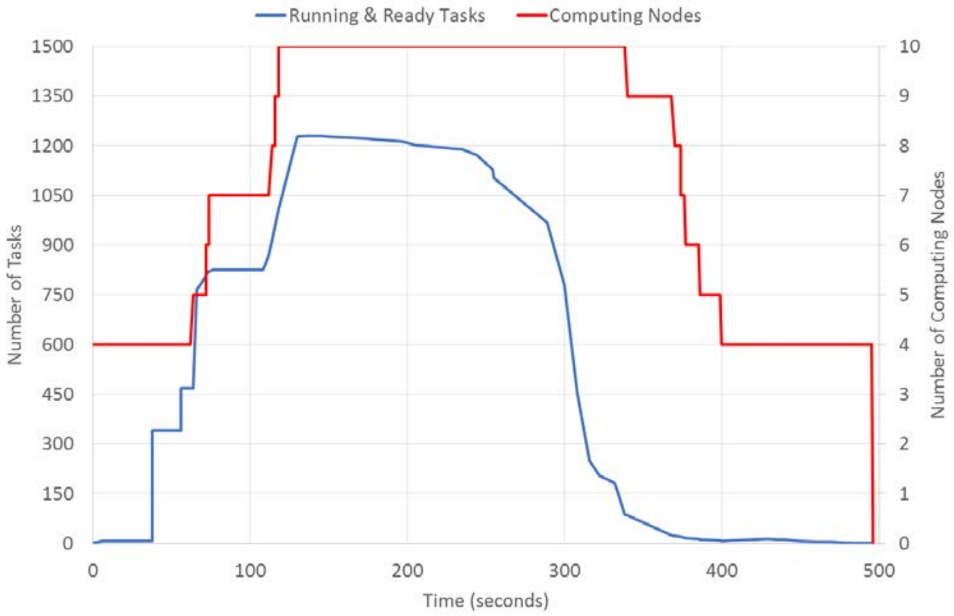


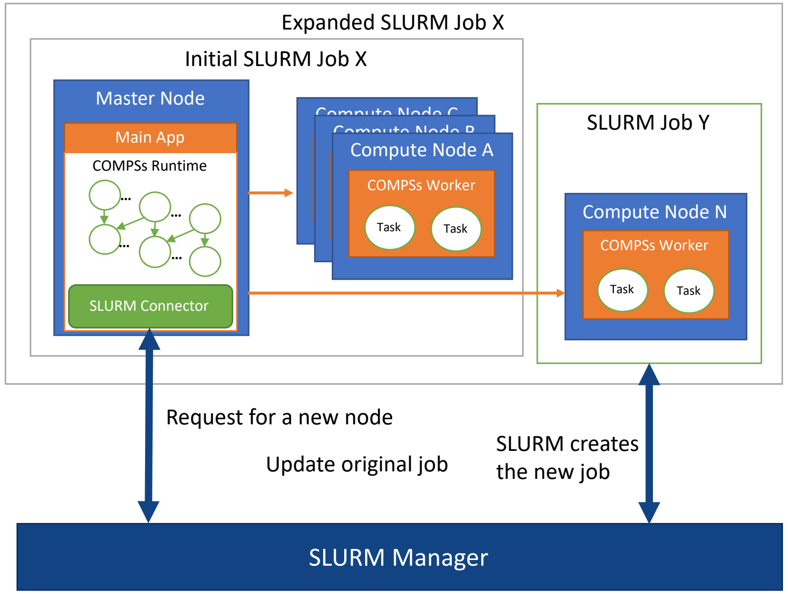


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

COMPSs runtime: support for elasticity

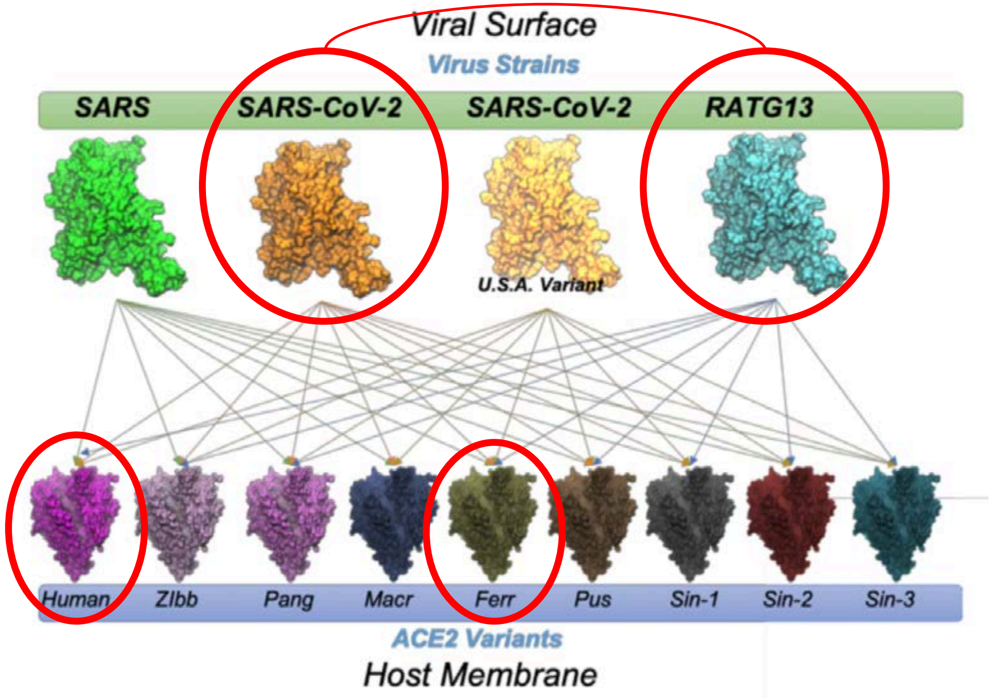
- Possibility to adapt the computing infrastructure depending on the actual workload
- Typical for cloud, now also SLURM managed systems
- Feature that contributes to a more effective use of resources
- Is **very relevant in the edge**, where power is a constraint

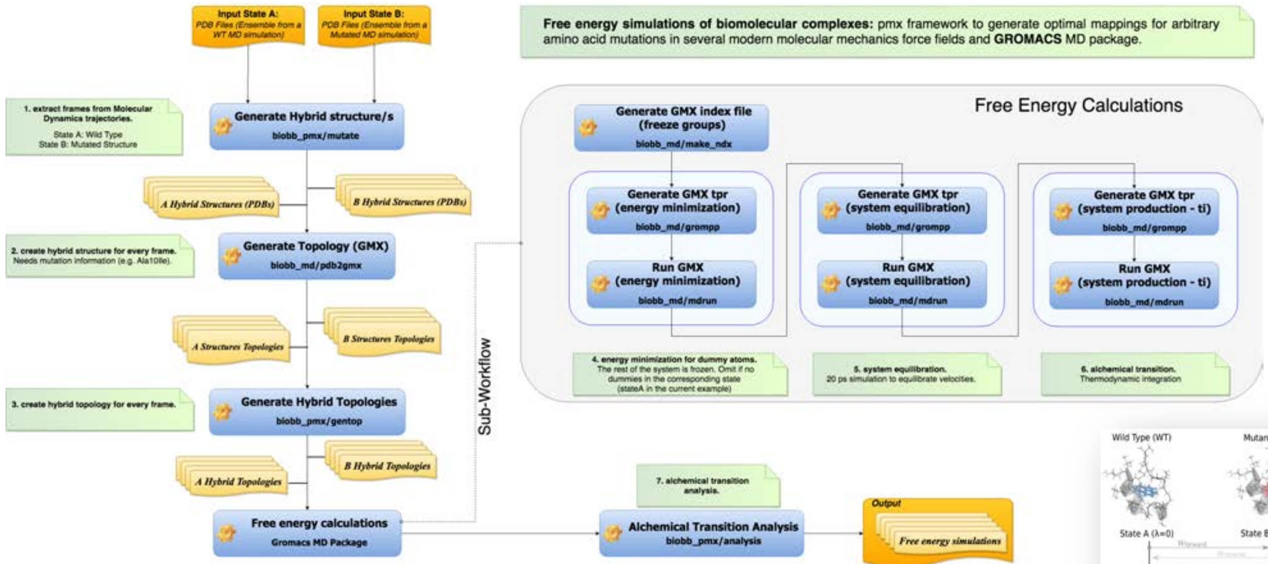




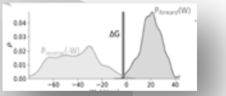
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

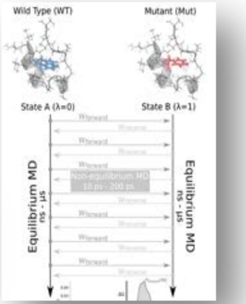
Pre-exascale workflows to explore COVID-19 Infectious Mechanisms and Host Selection Process





Impact of mutations in binding affinity





- **Fast-growth Thermodynamic Integration**
- **1000** independent **short MD simulations** (500 forward + 500 reverse)
- **GROMACS + pmx**
- **Extremely parallelizable**

GROMACS
FAST. FLEXIBLE. FREE.

pmx: generate hybrid protein structure and topology
Computational Biomolecular Dynamics Group

Molecular Dynamics simulation data

- For each mutation:
  - MD Simulations (RBD + ACE2 + Complex)
  - Free energy calculations

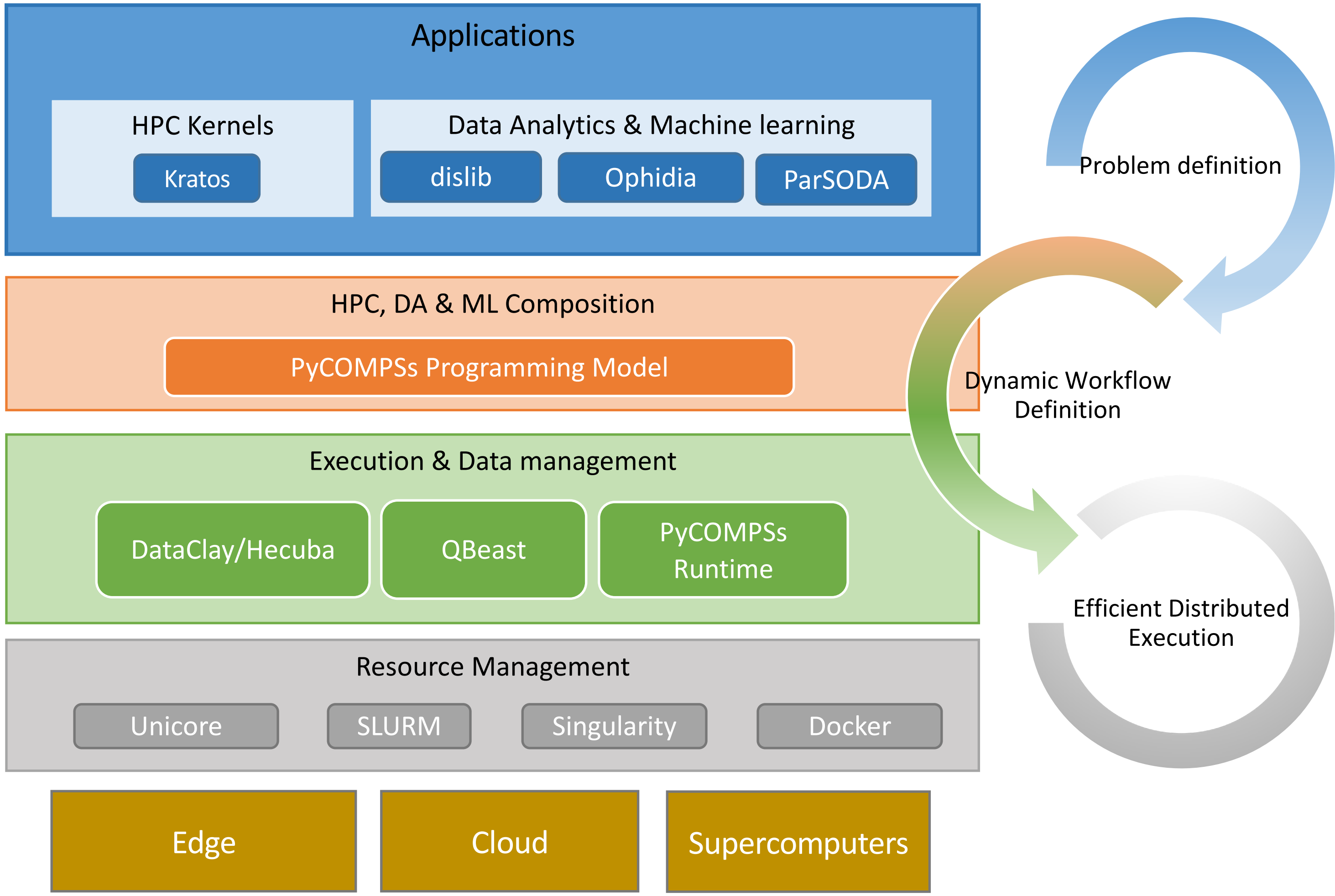
PRACE

Big Data Study:
PRACE computational power
Massive amount of data

bioexcel biobb pyCOMPSs BSC IRB BARCELONA

Conclusions

- PyCOMPSs 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 involved

bioexcel
Center of Excellence for Computational Biomolecular Research

LAND SUPPORT

CLASS

ExaQUte
Exascale Quantification of Uncertainties for Technology and Science Simulation

ELASTIC

EXPERTISE

FUJITSU

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

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

- 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-ubuntu16/
- Applications

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

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

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Thank you

daniele.lezzi@bsc.es