Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Enabling the convergence of HPC and Data Analytics in highly distributed computing infrastructures

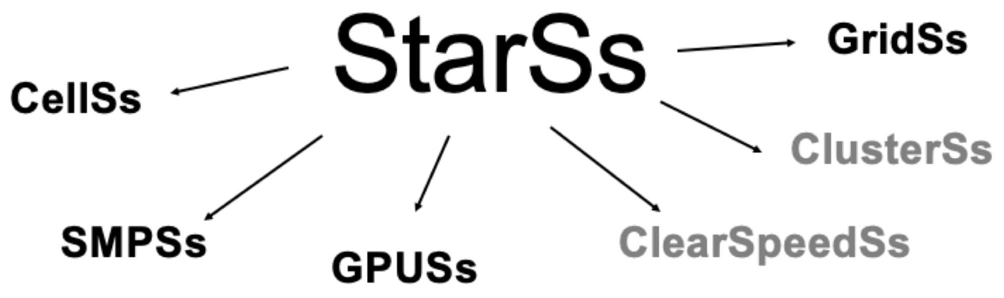
Rosa M Badia

1-2 July 2019

Yale: 80 in 2019, Barcelona

What was I doing when I first met Yale?

StarSs

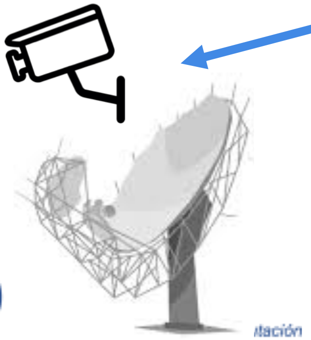


- StarSs
  - A "node" level programming model
  - C/Fortran + directives
  - Nicely integrates in hybrid MPI/StarSs
  - Natural support for heterogeneity
- Programmability
  - Incremental parallelization/restructure
  - Abstract/separate algorithmic issues from resources
  - Disciplined programming
- Portability
  - "Same" source code runs on "any" machine
    - Optimized task implementations will result in better performance.
  - "Single source" for maintained version of a application
- Performance
  - Asynchronous (data-flow) execution and locality awareness
  - Intelligent Runtime: specific for each type of target platform.
    - Automatically extracts and exploits parallelism
    - Matches computations to resources

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

AI everywhere

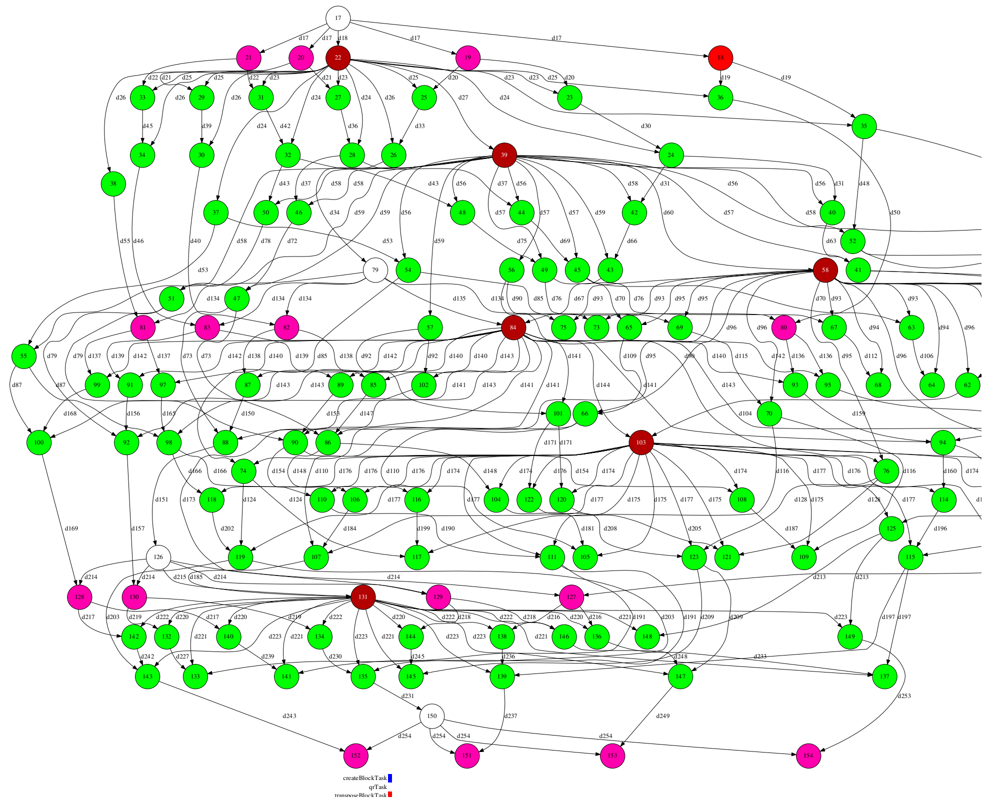


Data and storage challenge

- 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

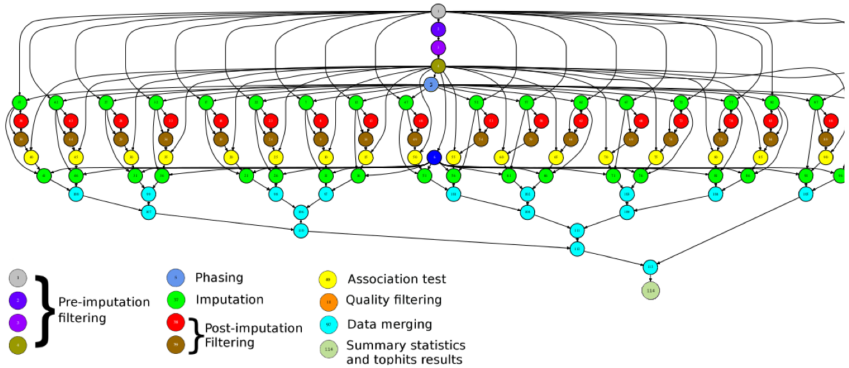
- How to describe the workflows in such environment? Which is the right interface?
- Focus:
  - Integration of computational workloads, with machine learning and data analytics
- Intelligent runtime that can make scheduling and allocation, data-transfer, and other decisions



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

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

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**

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

```
@constraint (MemorySize=1.0, ProcessorType ="ARM", )
@task (c=INOUT)
def myfunc_in_the_edge (a, b, c):
     ...
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

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 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")
@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):
    ...
```

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
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Failure management

- Default behaviour till now:
  - On task failure, retry the execution a number of times
  - If failure persists, close the application safely
- New 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
- **Offers the possibility of task speculation on the execution of applications**
- **Possibility of ignoring part of the execution of the workflow, for example if a task fails in an unstable device**

Integration with persistent memory

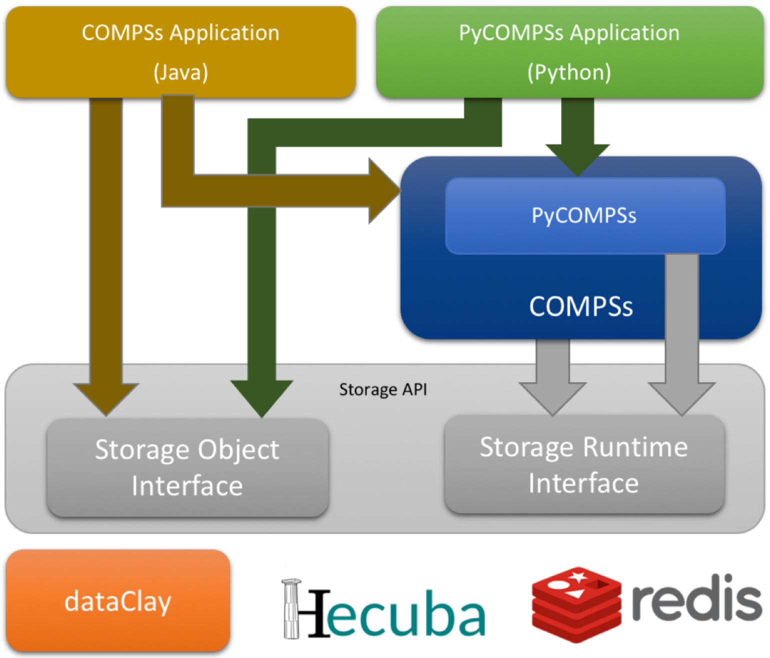
- Programmer may decide to make persistent specific objects in its code
- Persistent objects are managed same way as regular objects
- Tasks can operate with them

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

a.mytask()
compss_barrier()

o = a.another_object
```

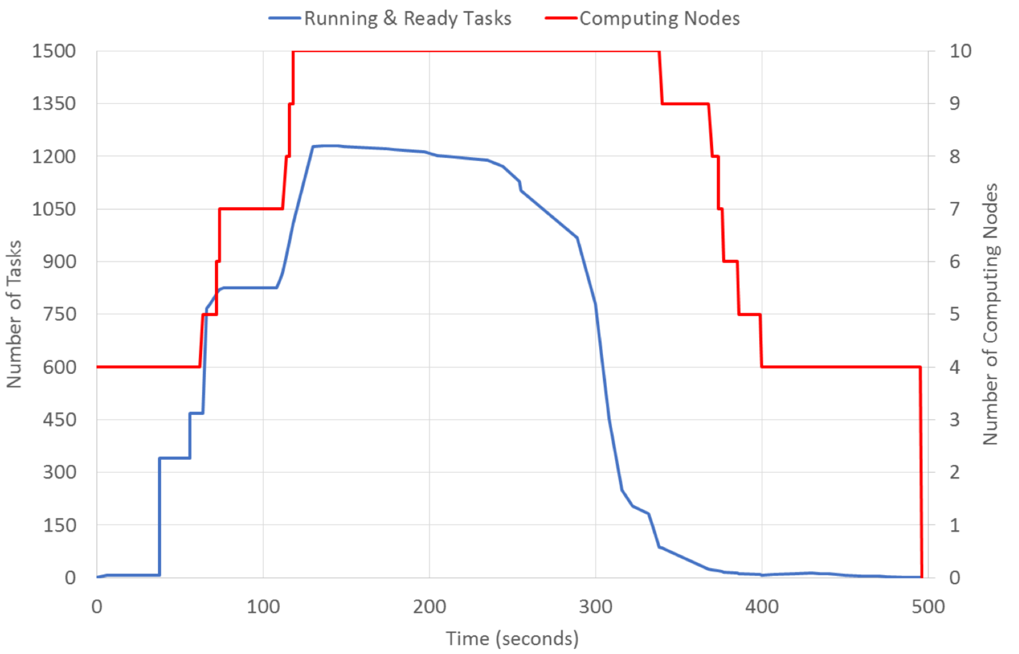
- **Objects can be accessed/shared transparently in a distributed computing platform**

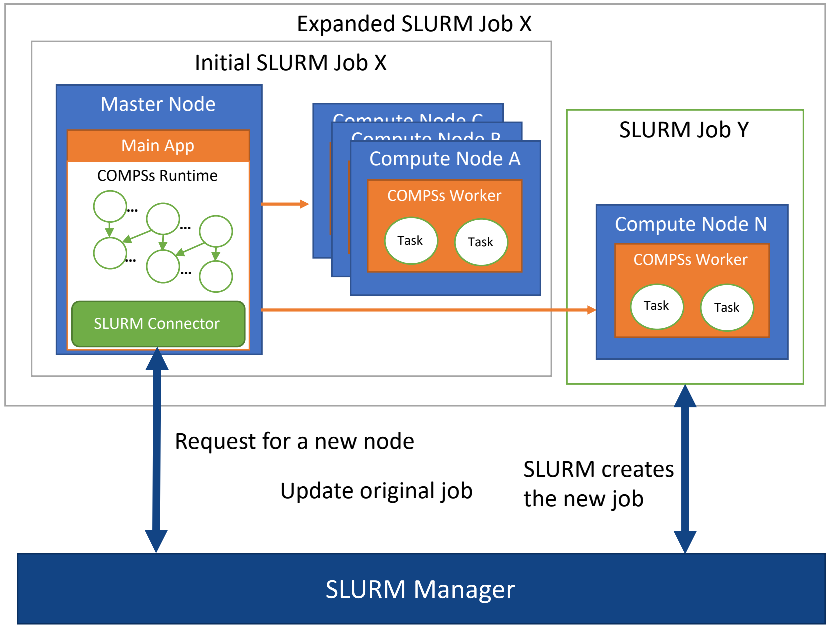


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

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
- Is **very relevant in the edge**, where power is a constraint

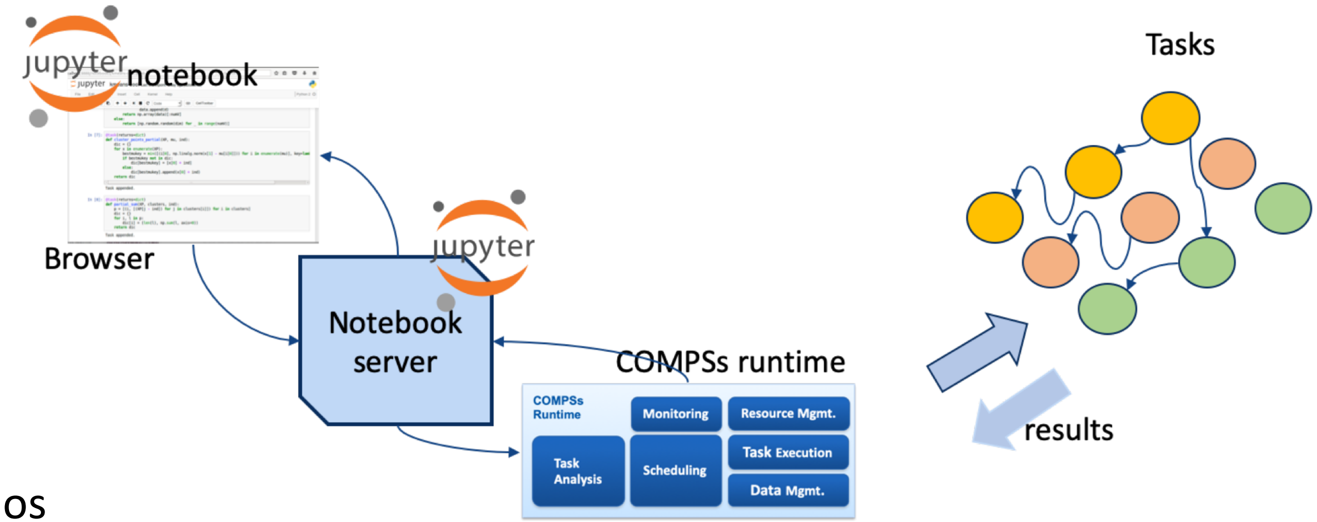




Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Support for interactivity

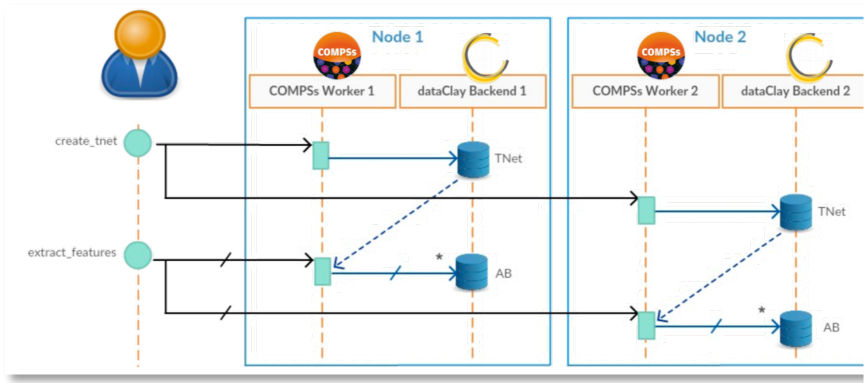
- Jupyter notebooks:
  Easy to use interface for interactivity



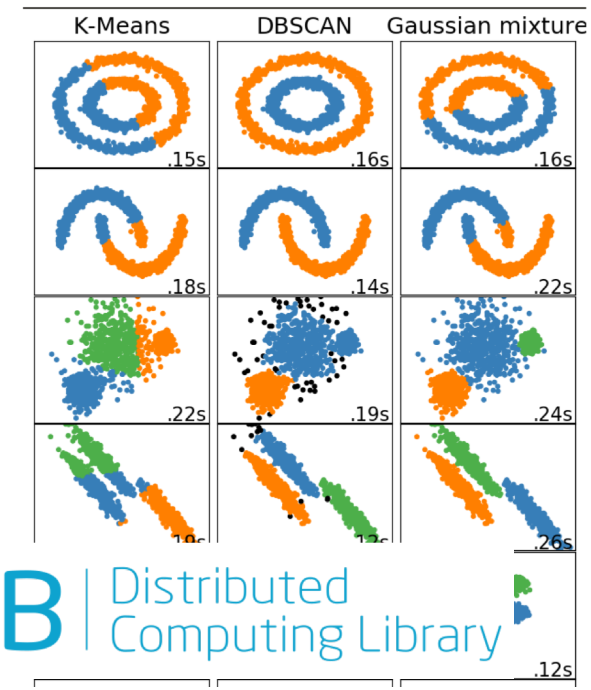
- Where to map every component?
  - Everything local
    - Prototyping and demos
  - Running notebook and COMPSs runtime locally
    - Some tasks can be executed locally
    - Some tasks can run remotely
      - Data acquisition in edge devices
      - Remote execution of compute intensive tasks in large clusters
  - Run browser in laptop and the notebook server and COMPSs runtime in a remote server
    - Enables the interactive execution of large computational workflows
    - Issue with large HPC systems if login node does not offer remote connection
    - Smoother integration if JupyterHub available

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



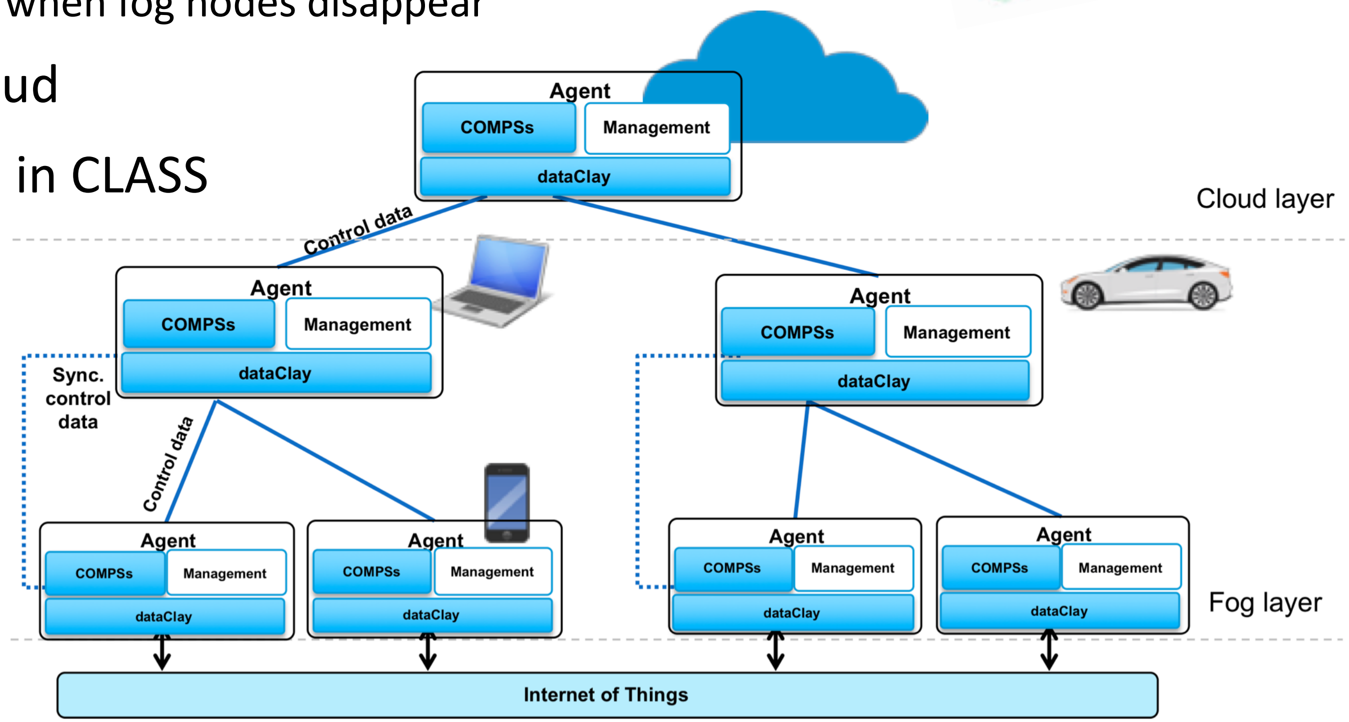
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

dislib.bsc.es

DISLIB Distributed Computing Library

COMPSs in a fog-to-cloud architecture

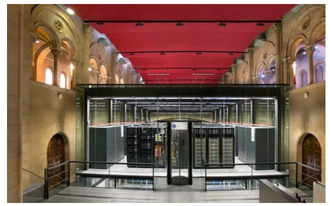
- **Decentralized** approach to deal with large amounts of data
- New COMPSs runtime handles distribution, parallelism and heterogeneity
- Runtime deployed as a microservice in an agent:
  - Agents are independent, can act as master or worker in an application execution, agents interact between them
  - 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
- Developed in mF2C, used in CLASS and ELASTIC



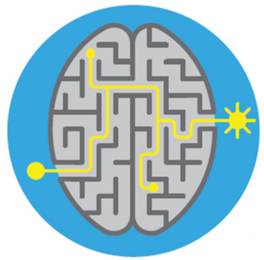
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Going beyond: what is missing

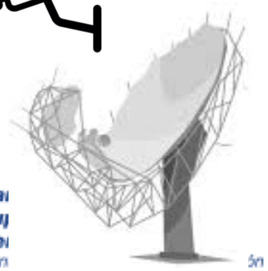
- Programming interfaces:
  - Explore graphical or higher-level interfaces to describe the workflows
- How to better integrate the compute and data flows
  - Integrate metadata, enable data traceability
  - Streaming
- Better support for interactivity, data-steering
- Add more intelligence to the runtime
  - Support for mapping sensors and actuators
  - Not only performance aspects, resilience and energy efficiency
  - Use of machine learning



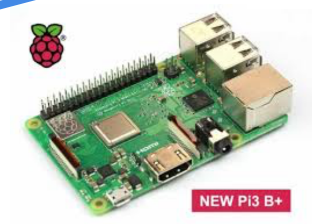
HPC
Exascale computing



AI



Sensors
Instruments
Actuators



Edge devices

pyCOMPSs

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-ubuntu16/
- Applications

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

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

Projects where COMPSs is involved

mF2C

LAND SUPPORT

CLASS

bioexcel
Center of Excellence for Computational Biomolecular Research

ExaQUte
Exascale Quantification of Uncertainties for Technology and Science Simulation

ELASTIC

EXPERTISE

Joint Laboratory for Extreme-Scale Computing

FUJITSU

nextgenio

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Thanks!

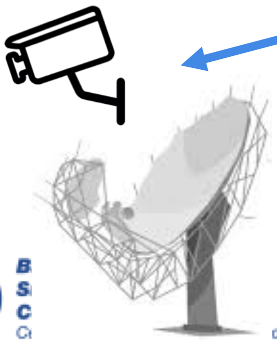
Challenges we are facing

- **Complex infrastructures**
    - Large number of nodes
        - Nodes that appear and disappear
    - Heterogeneous
    - Other relevant aspects: security and trust, power, ...
- Large amount of heterogeneous **data** from multiple sources. New storage technologies with different capabilities
- Need to **orchestrate** complex applications in such complex environment

HPC
Exascale computing
Cloud

Fog devices

Edge devices



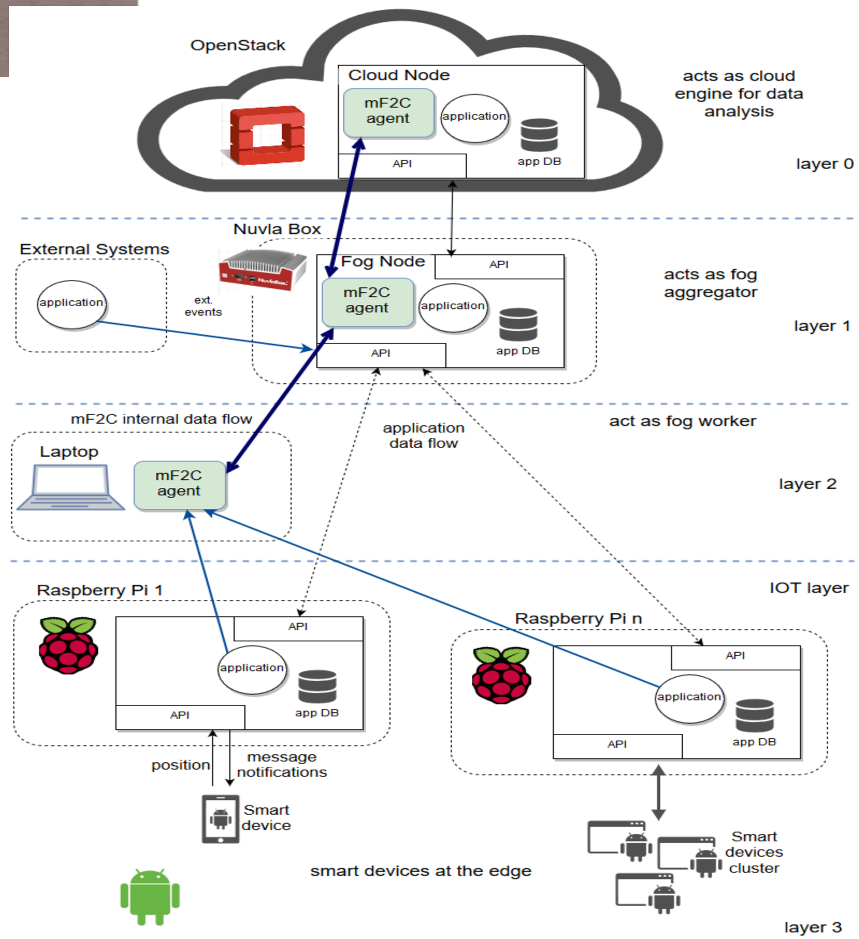
Sensors
Instruments
Actuators

mF2c - Smart Fog Hub System

Grant Agreement No 730929

- Indoor navigation and recommender solution at the Cagliari airport

Layer 0, cloud: OpenStack
Layer 1, fog aggregatot: Nuvla Box
Layer 2, fog: Laptop
Layer 3, IOT layer: Raspberry Pi, smartphones

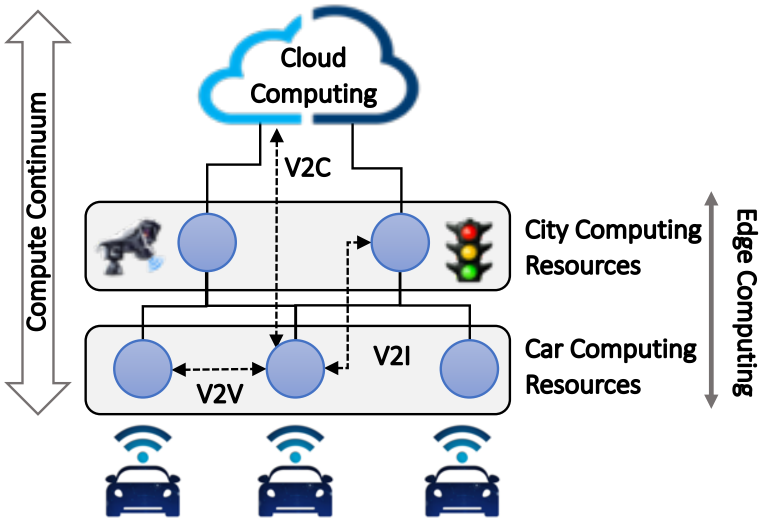


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Other use cases

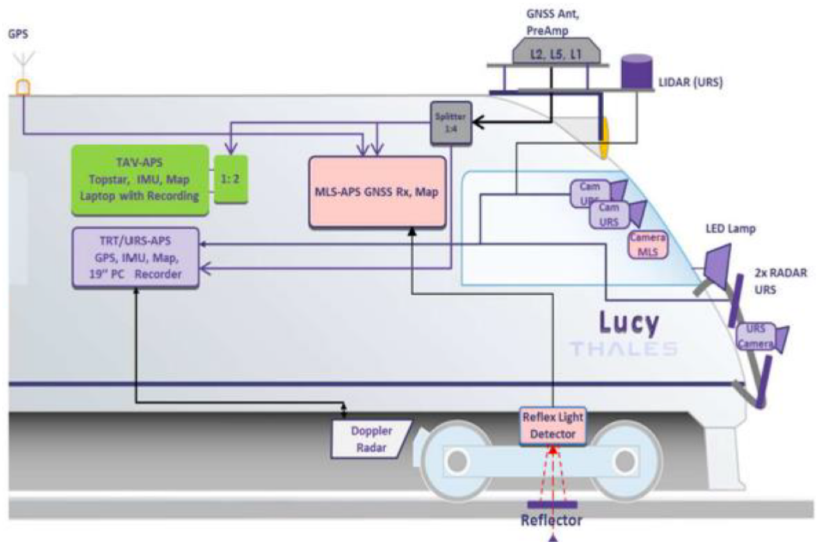
CLASS

- Intelligent traffic management
- Advanced driving assistance systems



ELASTIC

- Next Generation Autonomous Positioning (NGAP)
- Advanced Driving Assistant System (ADAS) (obstacle detection)
- Predictive maintenance

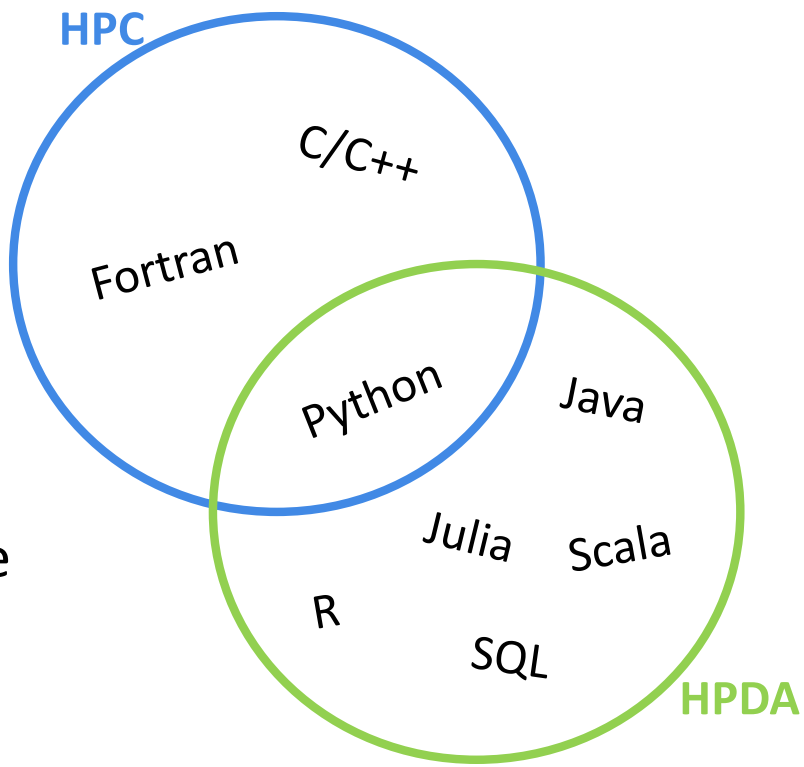


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

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



* From python.org