www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Programming the Cloud with PyCOMPSs: a task-based approach

7th Annual IMDEA Networks Workshop
Madrid, June 11th 2015

Human Brain Project

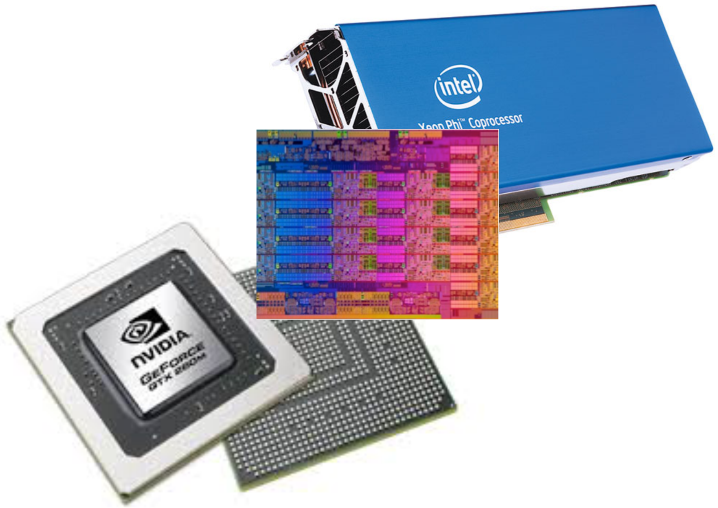
EXCELENCIA SEVERO OCHOA

Outline

- Motivation
  - Issues programming the cloud
  - BSC approach
  - Pillars
- BSC views on programming models
  - StarSs
  - PyCOMPSs
- Summary and projects

Computation platforms

- New architectures and organization of processors
  - Multicore
    - Including vector units
  - GPU/accelerators
  - FPGAs



- Shift on programming paradigms:
  - From sequential to parallel
  - New instructions/languages
- Computing paradigms:
  - From Clusters, through Grids, to Cloud

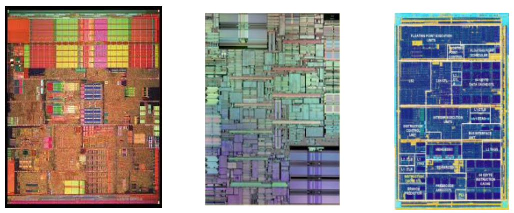
Application programming

Programs "decoupled" from computing platform

Applications

Simple interface
Sequential program

Programming language



Regular processors

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Programming evolution for distributed programming

- Distributed computing APIs make programming more complicated

Applications

Program logic
+
Platform specificities

Programming language + API



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

BSC vision on programming models

Applications

PM: High-level, clean, abstract interface

Power to the runtime

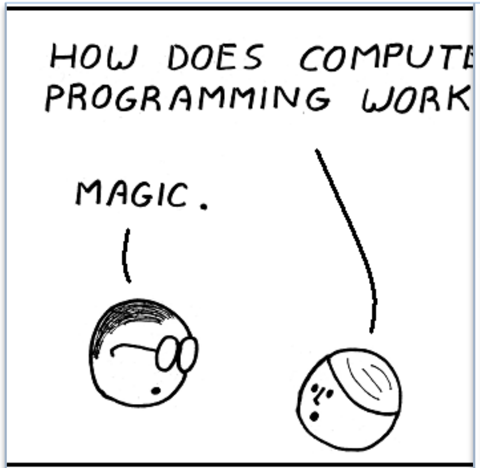
API

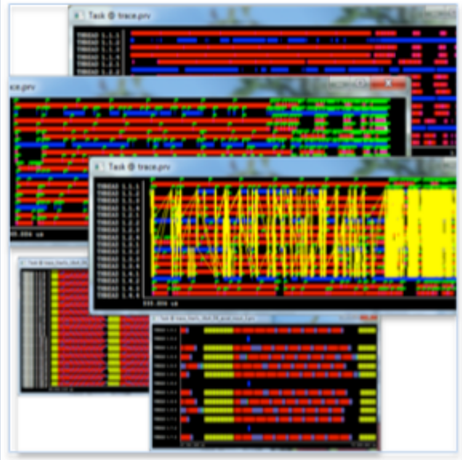
Program logic independent of computing platform

General purpose
Task based
Single address space

Intelligent runtime, parallelization, distribution, interoperability

Pillars for BSC strategy on programming models

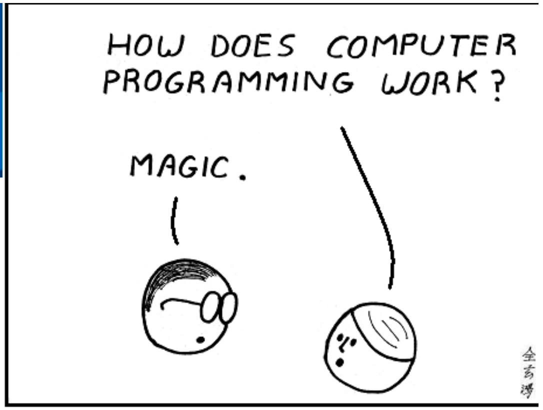




Programmability	Performance optimization	Portability	Efficient data access

Programmability

- Capability of being programmable
- …but good programmability of a programming model refers to
  - Easy to be used to develop applications
  - Easy to be read, good expressivity
  - More semantics with less lines of codes
- Sequential programming
  - Maybe we can think in parallel, but we communicate sequentially
  - One think at a time, do not need synchronization
  - Most programming languages are thought to be executed sequentially
- Parallel and distributed programming
  - The user must express parallelism, data distribution, and typically synchronization and communication
  - The user needs to manage data transfers between nodes
  - The population of users who can effectively program parallel and distributed is a small fraction

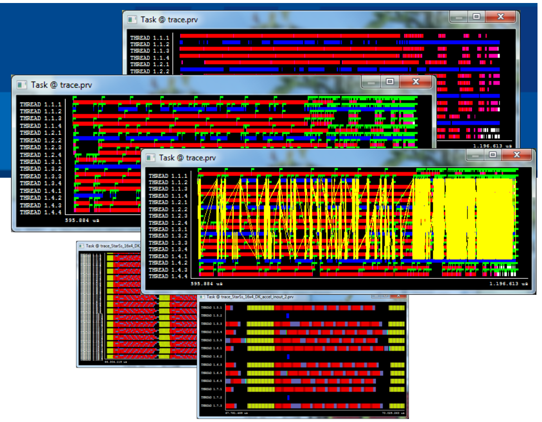


Portability

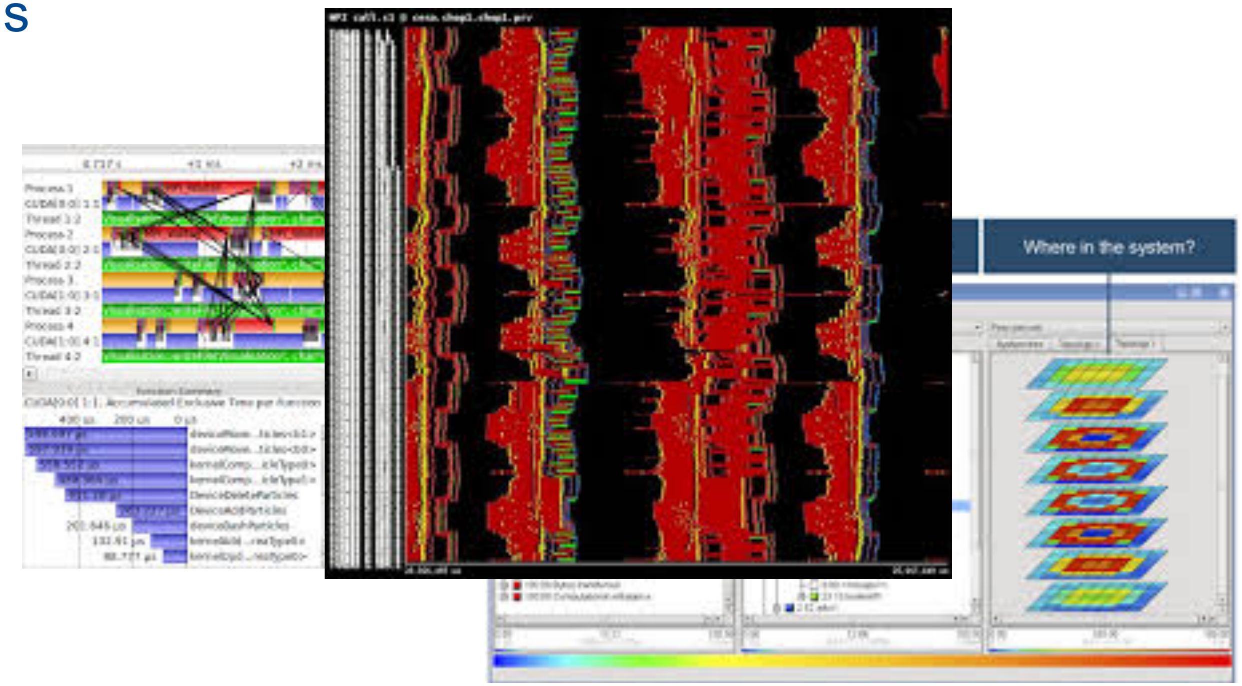
- Software portability
  - Measure of how easily an application can be executed in different computing environments
  - Requires generalized abstraction between the application logic and system interfaces
  - Key issue for development cost reduction
- A computer software application is considered portable
  - If the effort required to adapt it to the new environment is within reasonable limits

- Issues
  - New ISAs – Extensions to vector instructions
  - New Architectures - GPUs
  - In distributed environments: different middlewares
    - I.e., cloud APIs

Performance optimization



- Methodologies to make applications faster
- From sequential to parallel/distributed
- … but also
  - Vectorization
  - GPUs
- Methodologies to make applications more efficient
  - Performance analysis
  - Monitoring
  - Performance tuning



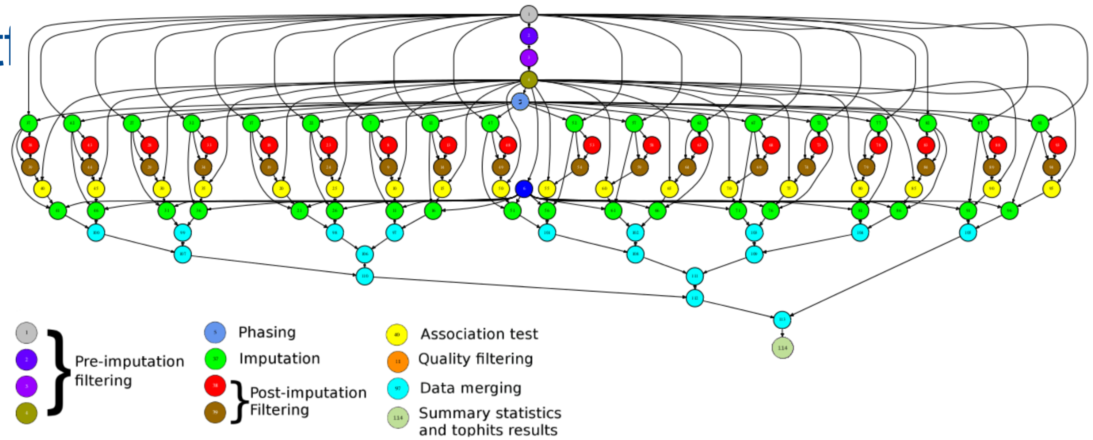
Data access revolution

- New storage devices
  - NVRAM
  - Storage Class Memories (SCM)
- Resemble more memory than storage
  - Low latency, high bandwidth, byte-addressable interface
  - Using them as block devices for a file system does not seem to be the best option
- Imply new storage methodologies
- May imply a disruption on how data is accessed

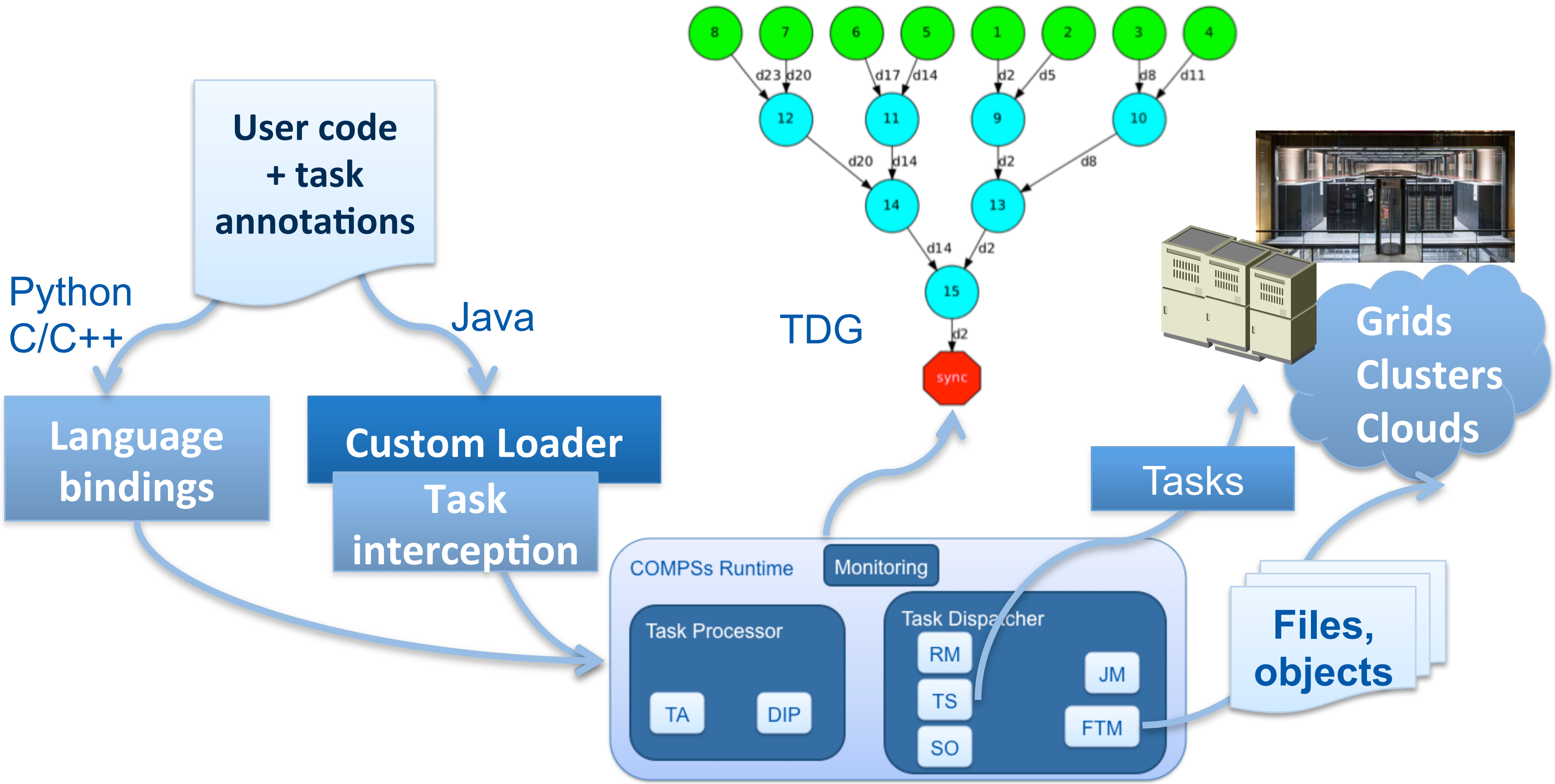
Programming with COMPSs

- 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
- Automatic on-the-fly creation of a task dependency graph
- Exploitation of parallelism
  - … and of distant parallelism
- Agnostic of computing plat
  - Enabled by the runtime for clusters, clouds and grids

Open Source
http://compss.bsc.es



COMPSs: how does it works?



Why Python?

- Python is powerful... and fast;
  plays well with others;
  runs everywhere;
  is friendly & easy to learn;
  is Open. *
- Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C
- Large community using it, including scientific and numeric
- Object-oriented programming and structured programming are fully supported
- Large number of software modules available (38,000 as of January 2014)

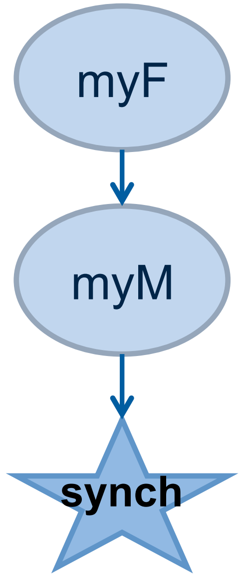
* From python.org

Python (PyCOMPSs) syntax

- Based on regular/sequential Python code
- Decorators to identify tasks
- Small API for data synchronization

Main Program

```
foo = Foo()
myFunction( foo )
foo.myMethod()
…
foo = compss_wait_on(foo)
foo.bar()
```



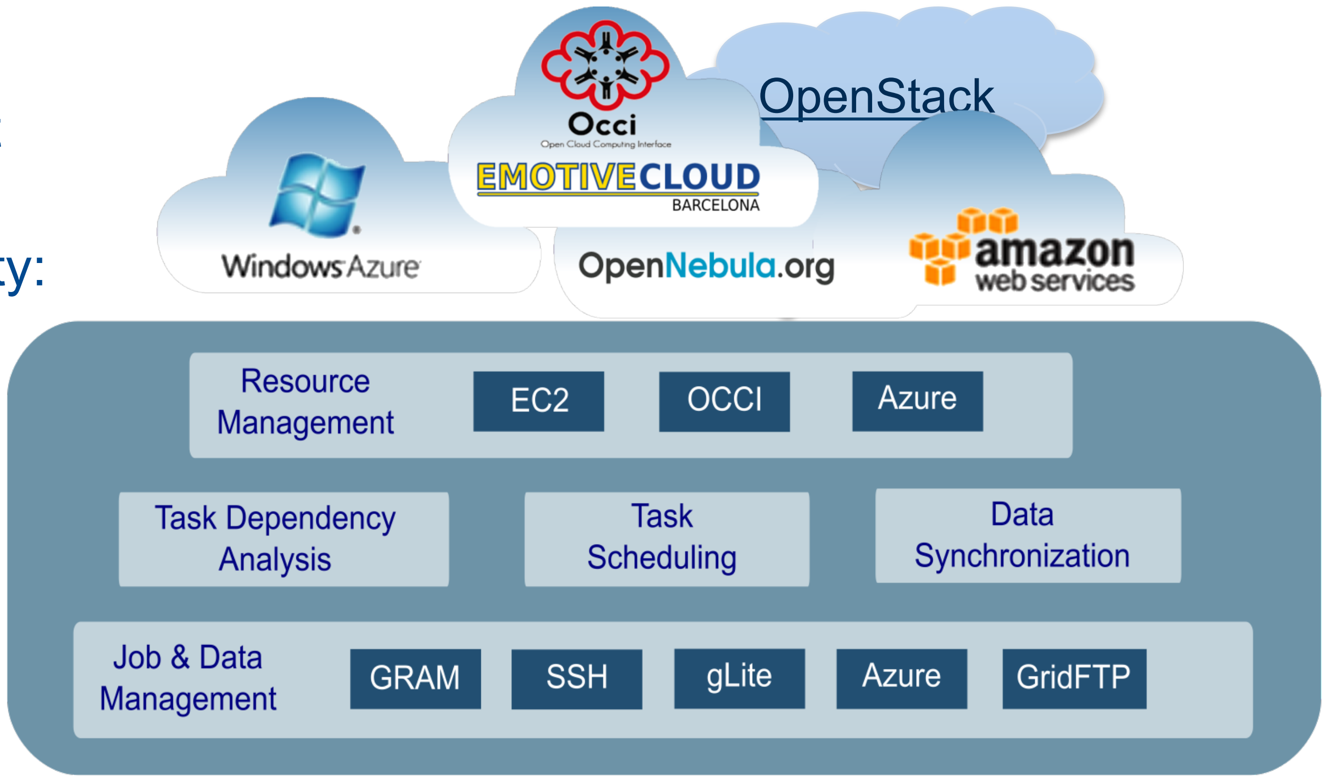
Function definition

```
@task( par = INOUT )
def myFunction(par):
  …
```

```
class Foo(object):
  @task()
  def myMethod(self):
    …
```

Runtime System

- Platform agnostic
- Support for different grid middlewares
- Cloud interoperability:
  - Public and private
  - Heterogeneous clouds

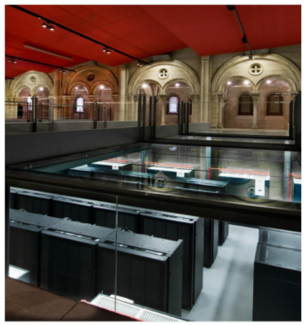


gLite Lightweight Middleware for Grid Computing

IBERGRID

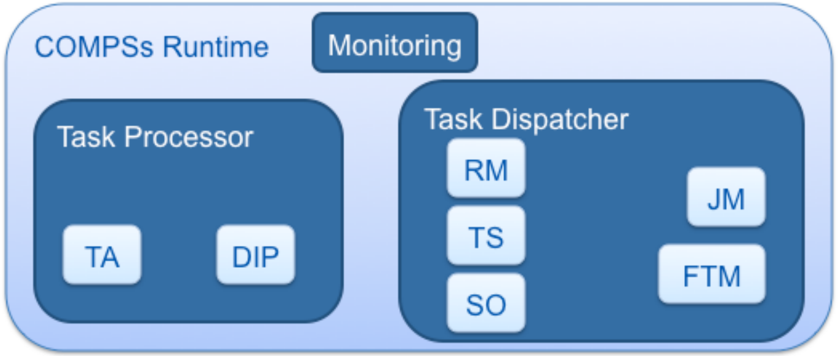
Open Science Grid

egi



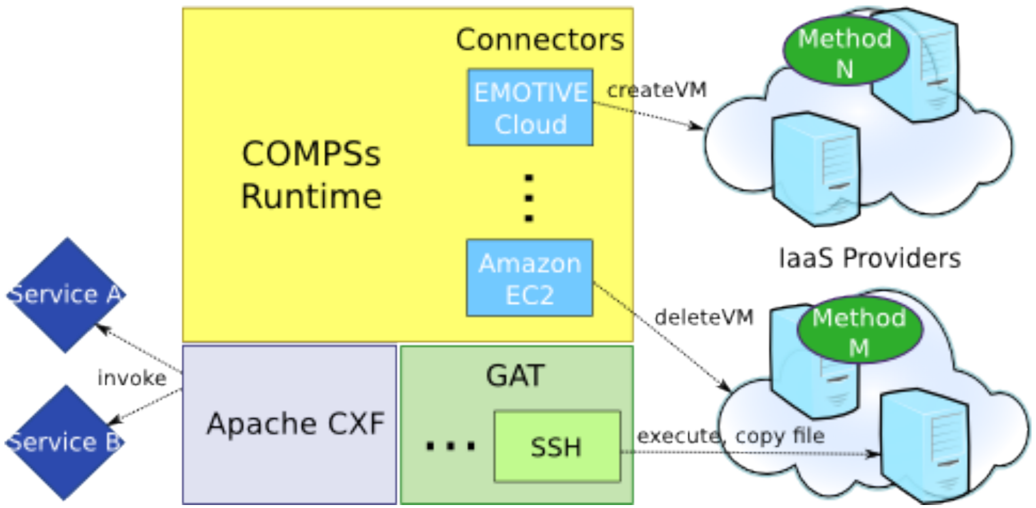
COMPSs Runtime: scheduling and resource management

- Task Scheduler
  - Assigns tasks to VMs or physical resources
  - Each VM or resource has its own task queue
- Scheduling Optimizer
  - Checks status of workers
  - Can decide
    - To perform load balancing
    - Create/destroy new VMs
- Resource Manager: elasticity
  - Manages all cloud middleware related features
  - Holds information about all workers and about cloud providers
  - Scheduler Optimizer sends to the RM requirements about new VM characteristics
  - Resource Manager, evaluates the cloud providers alternatives and chooses the best option
    - More economic
    - The decision can be to open a new private or public VM
  - For each Cloud provider, a data structure stores the different available instances (with its features) and the connector that should be used



Interoperability to cloud middleware through connectors

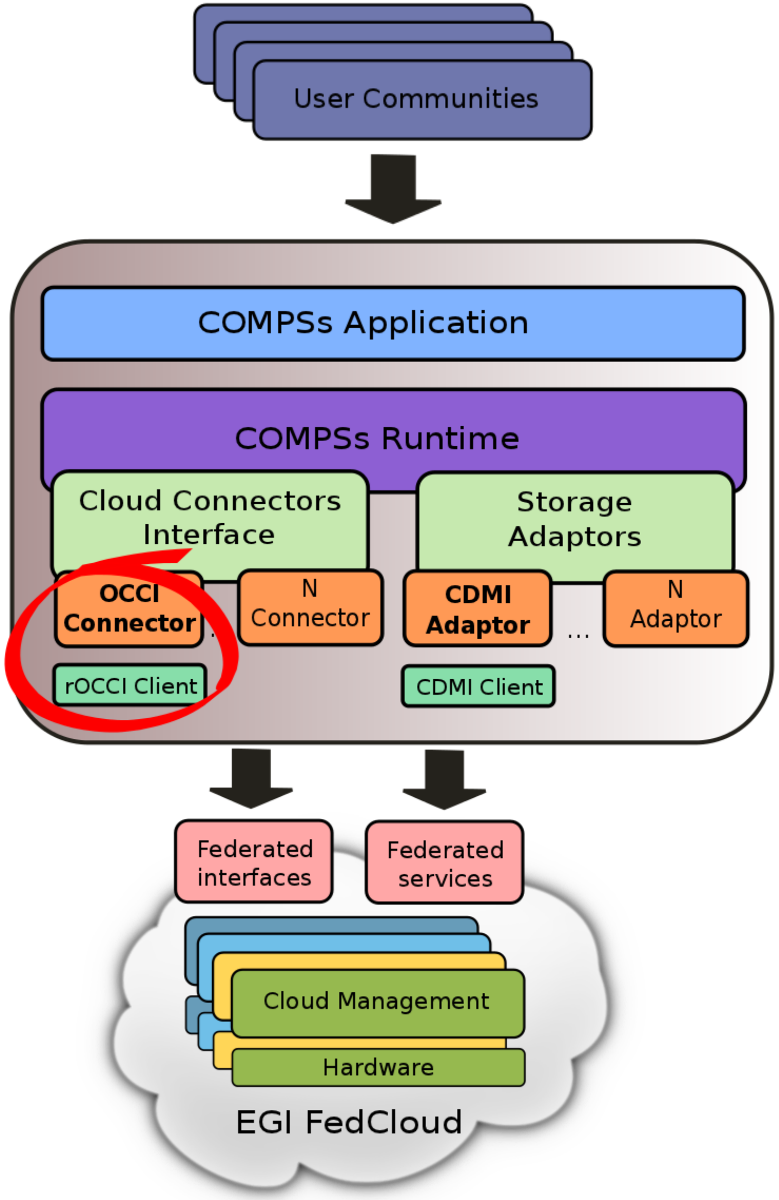
- The runtime communicates with the Cloud by means of Cloud connectors
- The connectors implement a common interface between the runtime and cloud provider
- Connectors abstract the runtime from the particular API of each provider
- This design facilitates the addition of new connectors for other providers
- Example:
  - Integration to EGI FedCloud through OCCI connector
- Available connectors
  - OpenNebula
  - OpenStack
  - Amazon



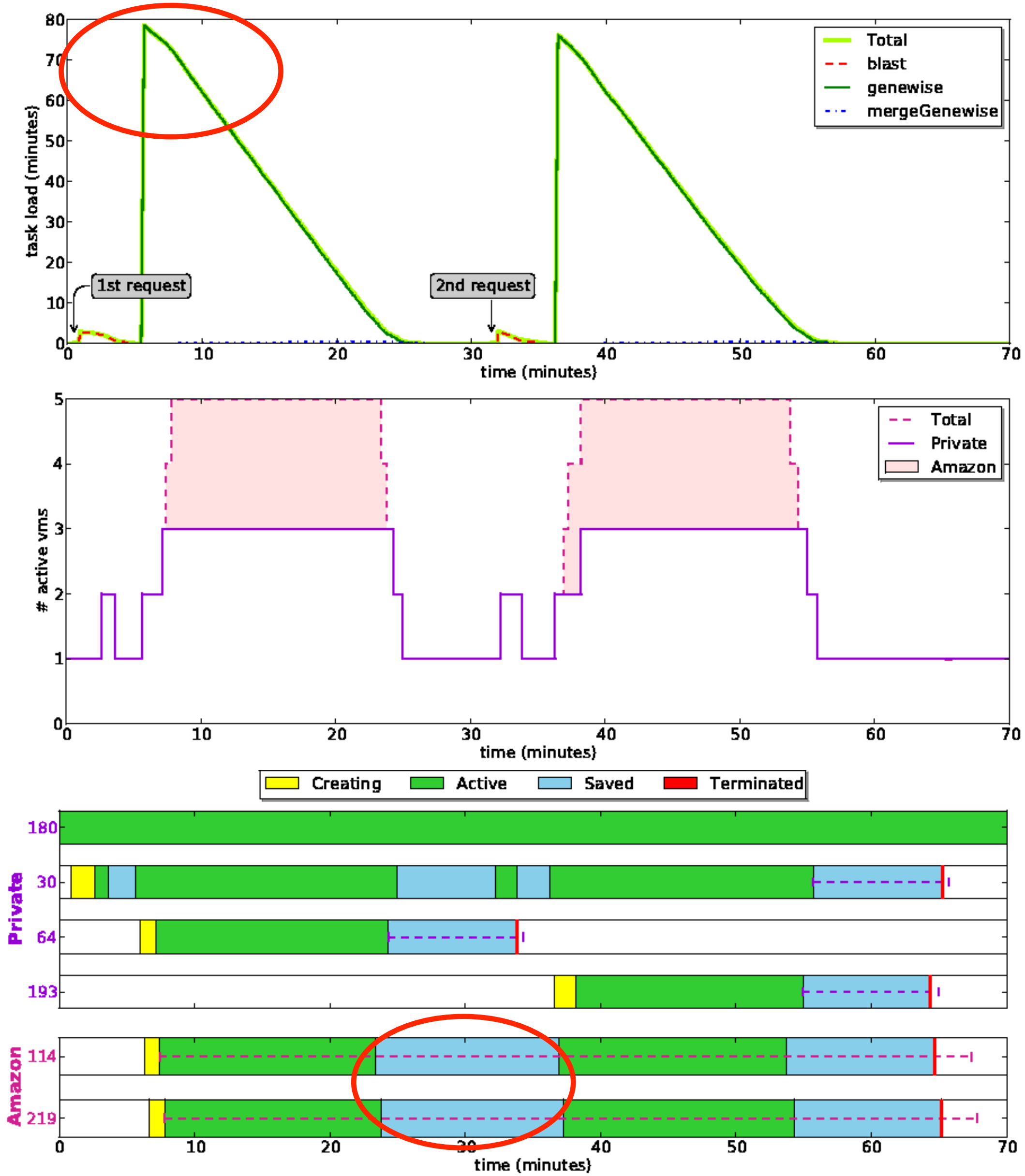
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

COMPSs integration with EGI FedCloud

- COMPSs Application: implementation of the application logic, where some tasks are executed remotely on EGI FedCloud resources
- Cloud Connectors
  - OCCI Connector: translates COMPSs resource management calls to OCCI operations.
- Different provider's configuration set up through COMPSs configuration files
- COMPSs available in the EGI software marketplace



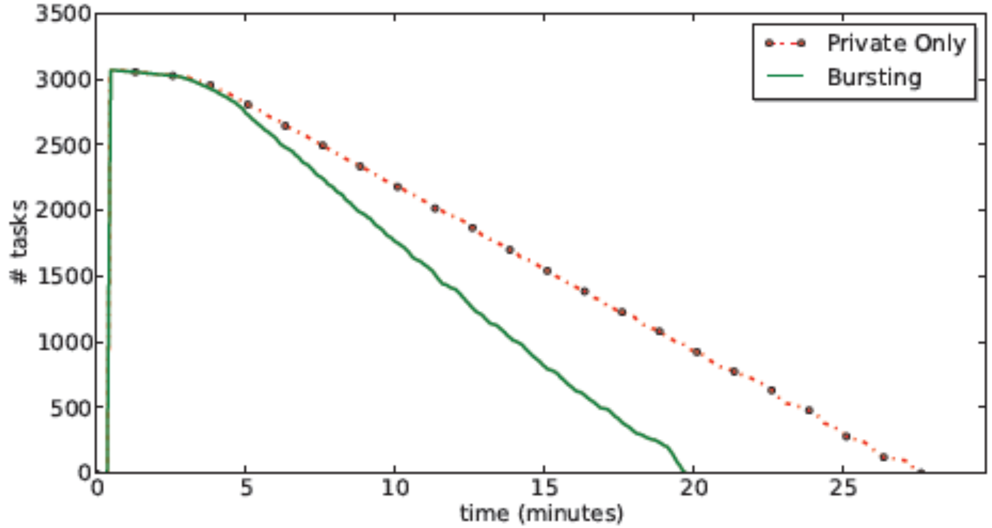
Elasticity in the Cloud

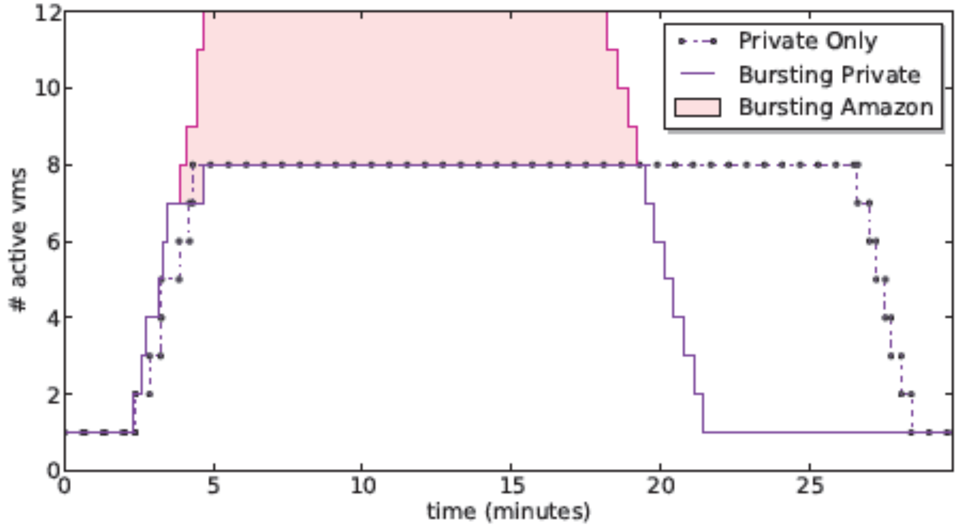


- Dynamic creation / destruction of VMs
  - Depending on task load
- Bursting to meet peak demands
  - Private Cloud (EMOTIVE)
  - Public Cloud (Amazon)
- Save VMs for later use
  - Amazon: use the whole hour slot
- Reuse of VMs
- VM deadlines

Elasticity in the Cloud

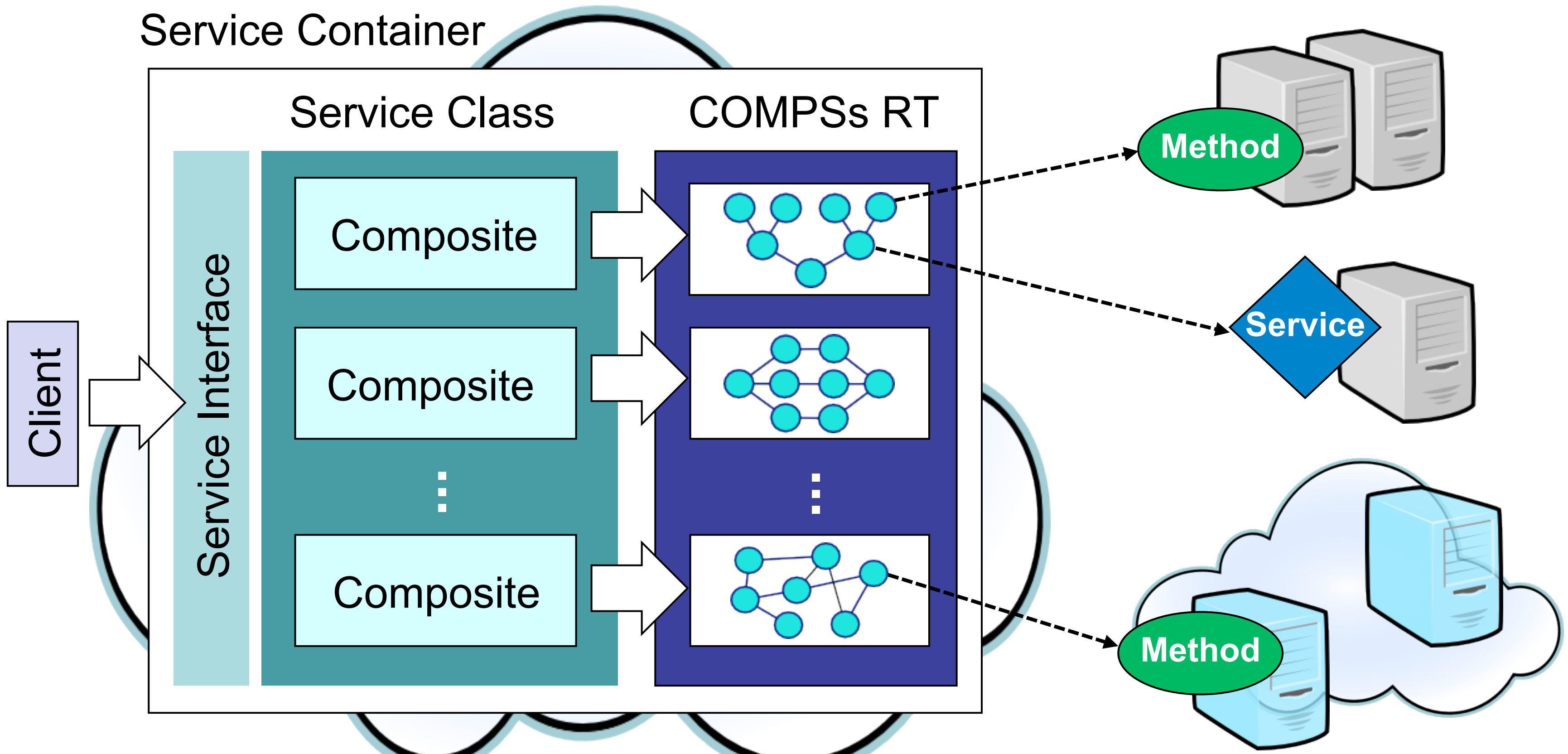
- Scalability
  - Private Cloud: the entire workflow in a single provider
  - Hybrid (Private + Public): tasks and data distributed over two distant providers





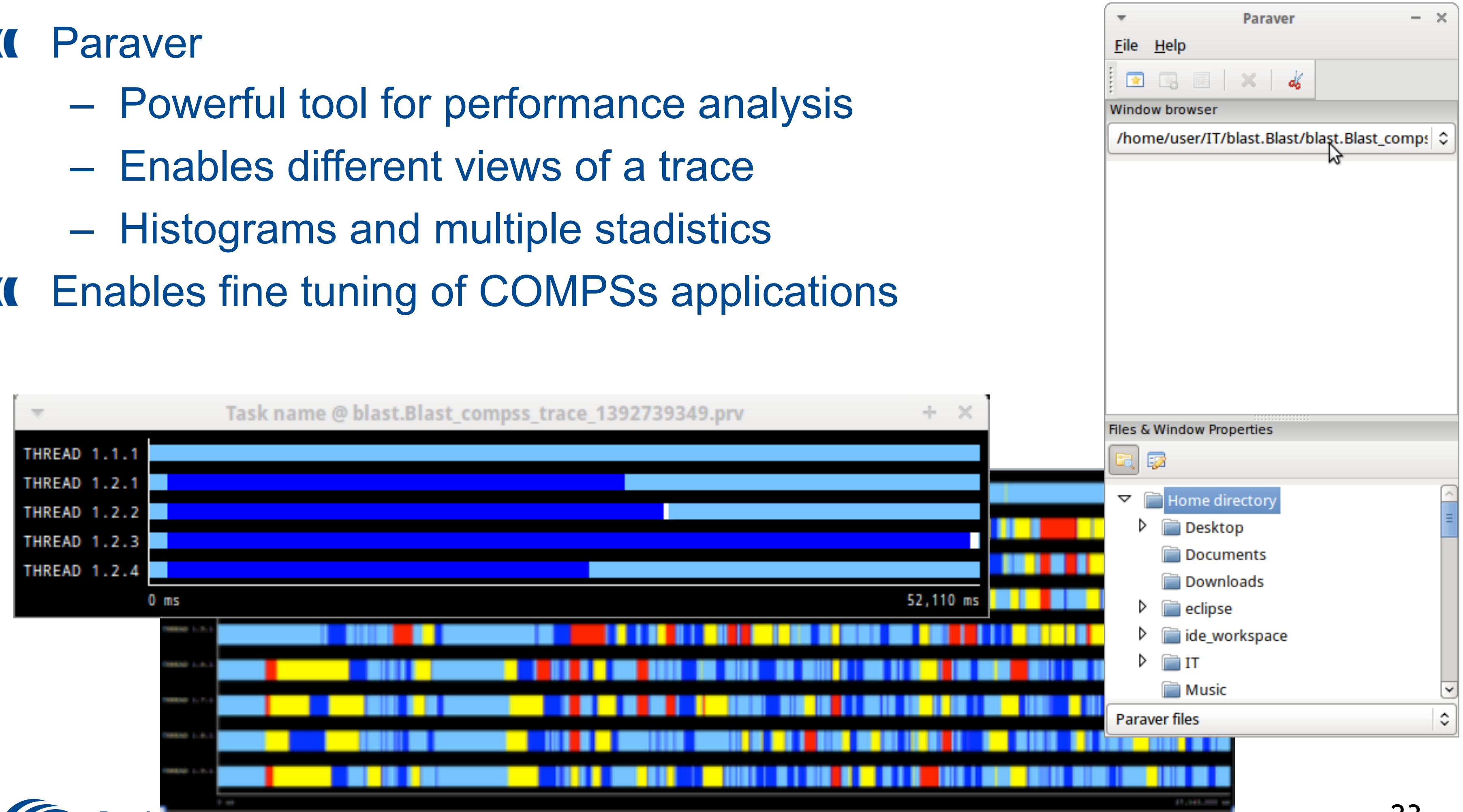
COMPSs to deploy SaaS

- Service orientation at two levels:
  - Specific COMPSs tasks can be services
  - COMPSs applications can be deployed as a service



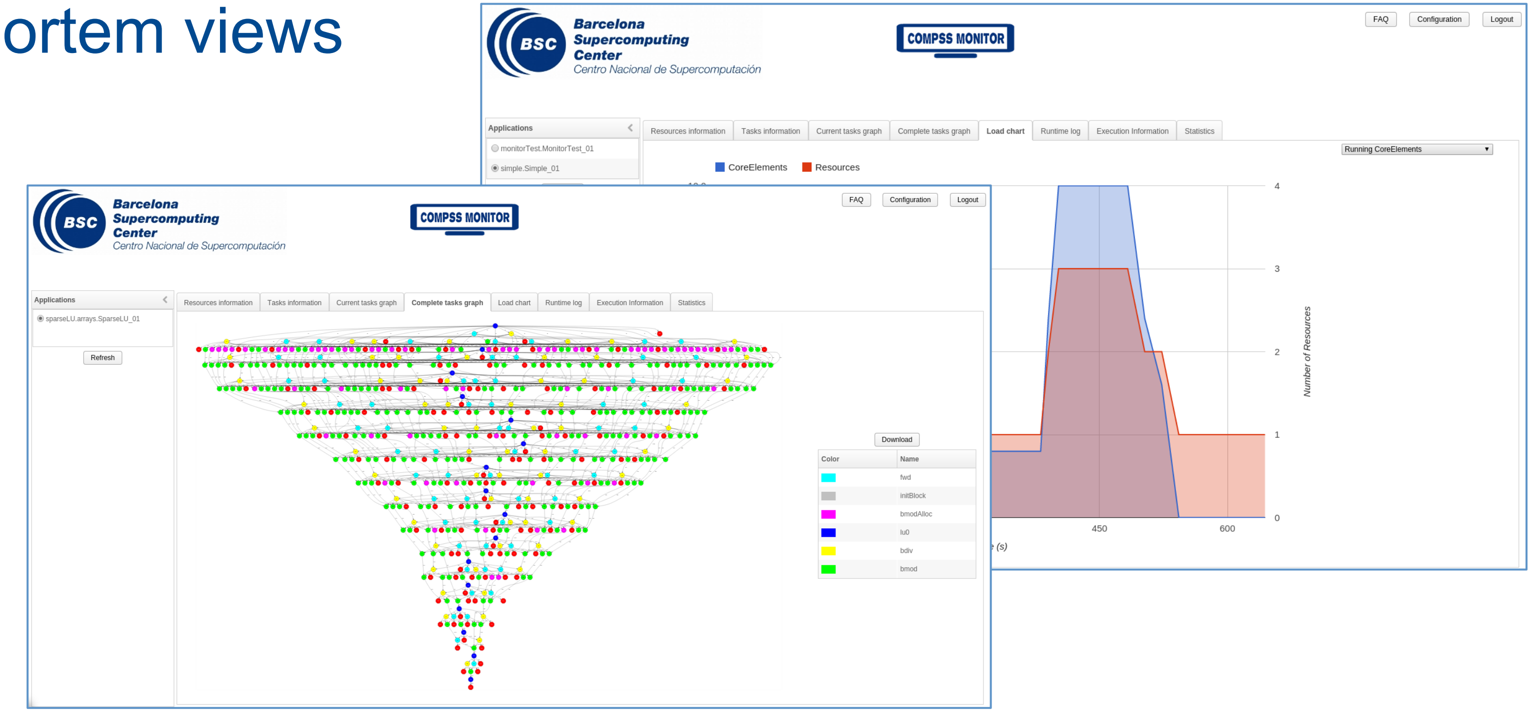
Performance optimization

- COMPSs runtime instrumented to generate post-mortem Paraver tracefiles
- Paraver
  - Powerful tool for performance analysis
  - Enables different views of a trace
  - Histograms and multiple stadistics
- Enables fine tuning of COMPSs applications



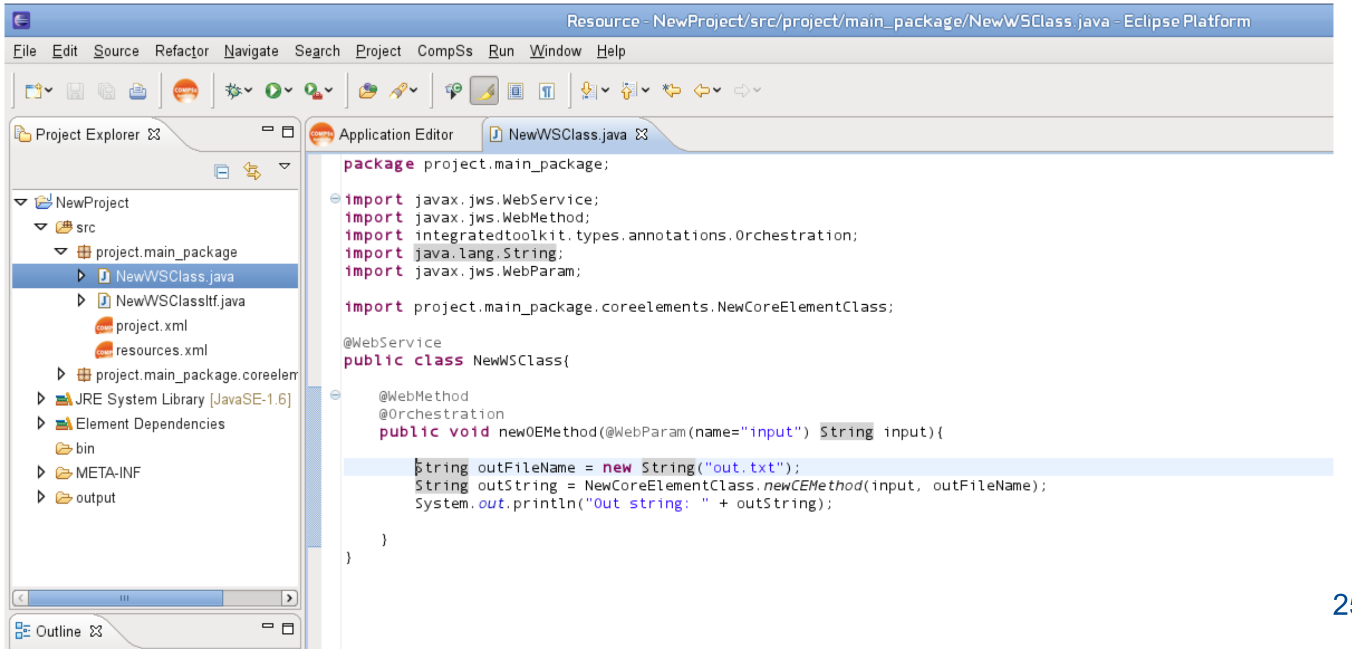
Performance monitoring

- Information collected at runtime about application
  - Task graph
  - Resources used
  - Workload
- Dynamic views at execution time
- Post-portem views

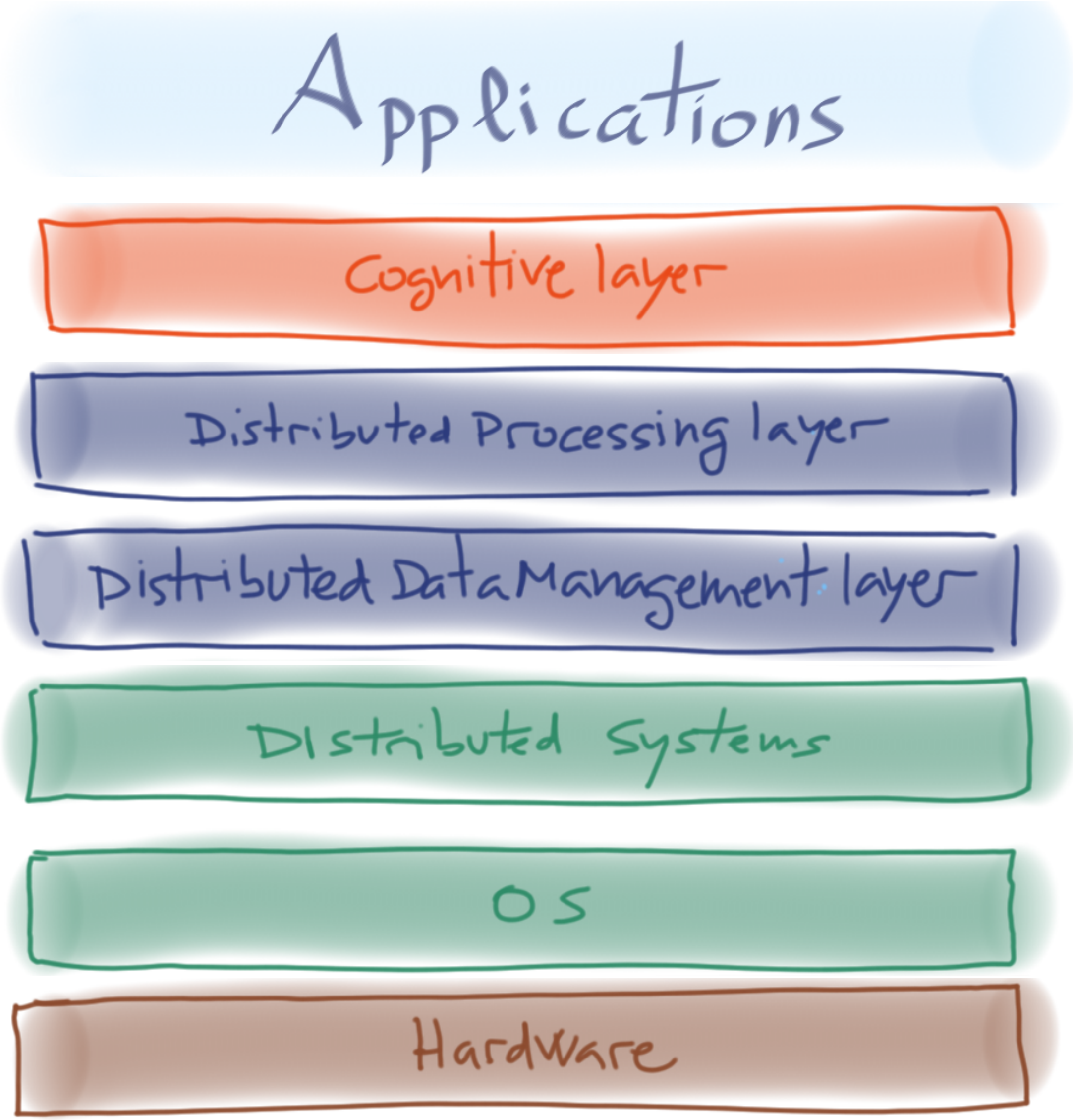


COMPSs IDE

- Graphical interface to help developers with COMPSs applications
  - Annotation of main program and tasks
  - Generation of project and resources files (xml)
  - Deployment in the infrastructure
- Developed as Eclipse plugin
  - Available in the Eclipse marketplace



Abstraction of computer middleware



e.g. Genomics

e.g. Machine Learning

e.g. MapReduce

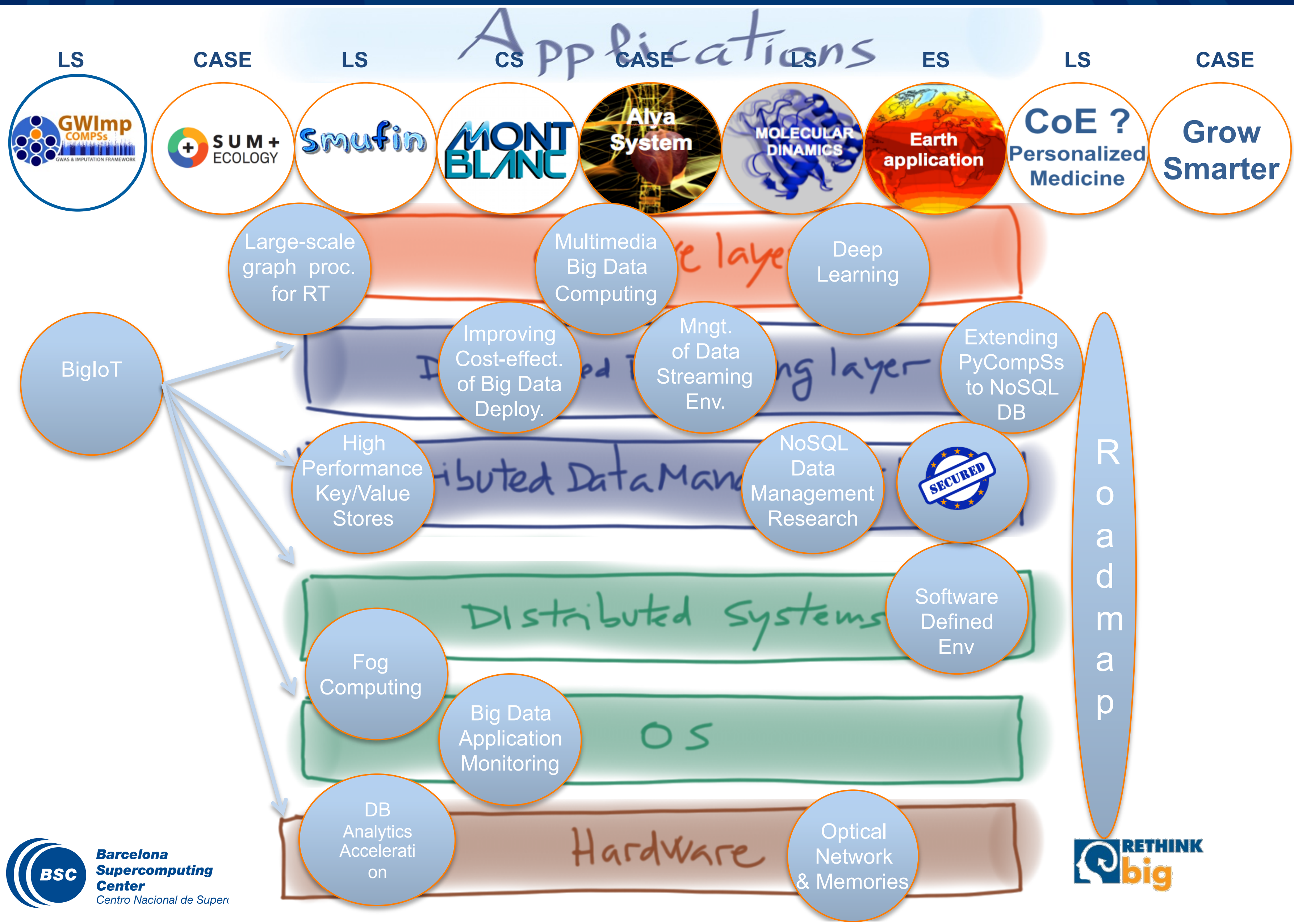
e.g. NoSQL DB

e.g. Cloud

e.g. Linux

e.g. FPGAs, GPUs

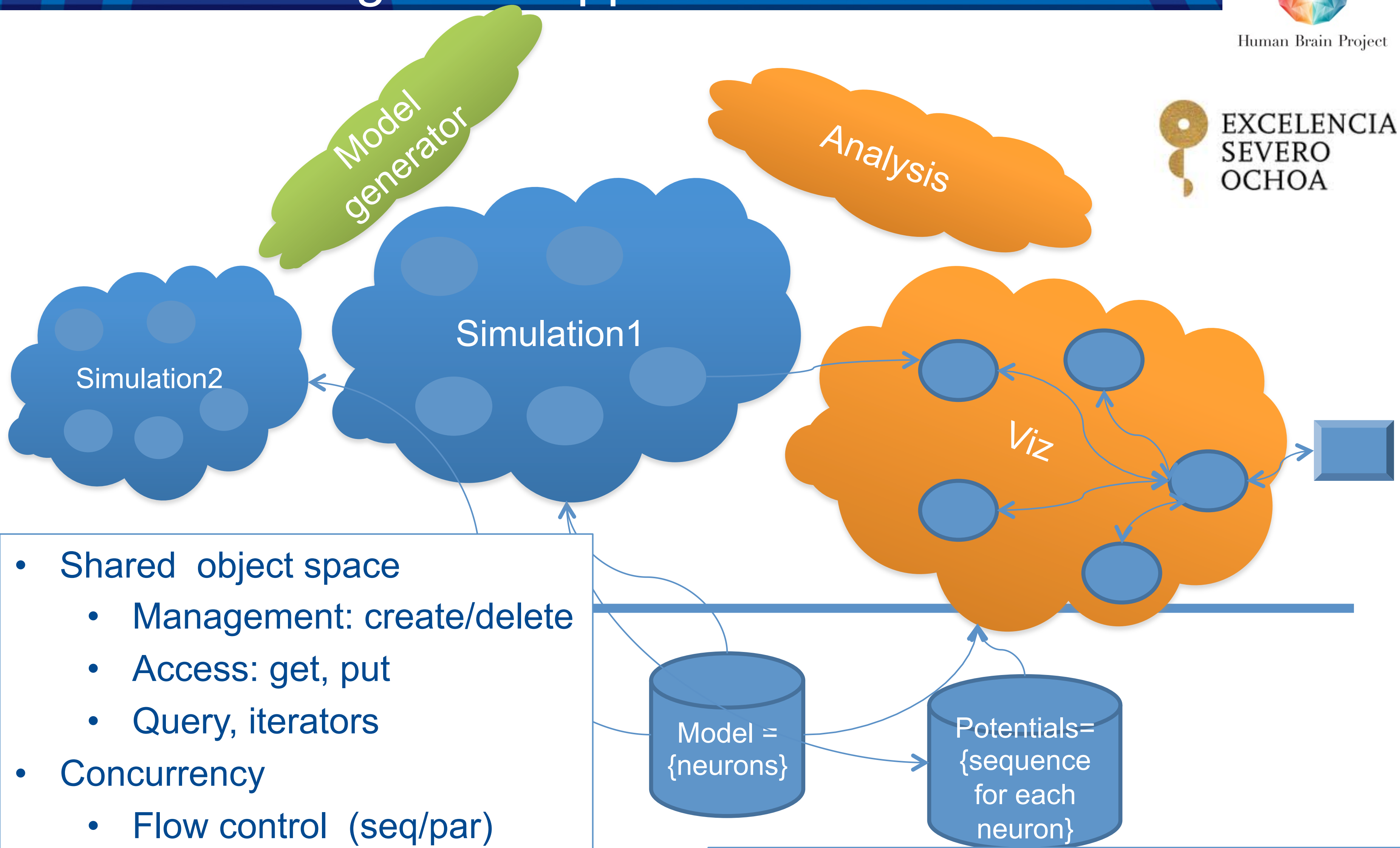
BSC Big Data related projects



COMPSs & Big Data: application scenarios

Human Brain Project

EXCELENCIA SEVERO OCHOA

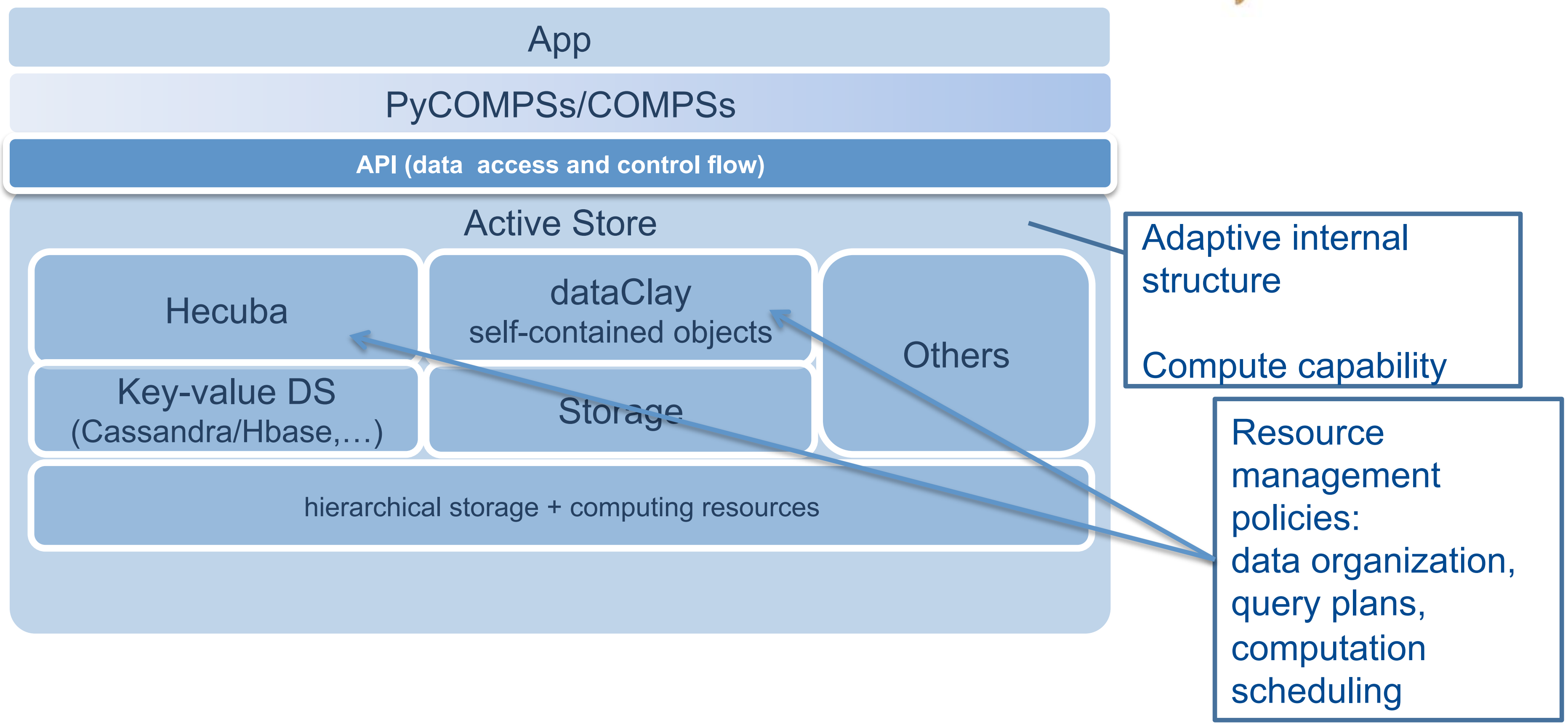


- Shared object space
  - Management: create/delete
  - Access: get, put
  - Query, iterators
- Concurrency
  - Flow control (seq/par)
  - Synchronization
- Consistency

- Implementation:
  - Persistent, Distributed, Resilient

PyCOMPSs integration with Big Data

Architectural design

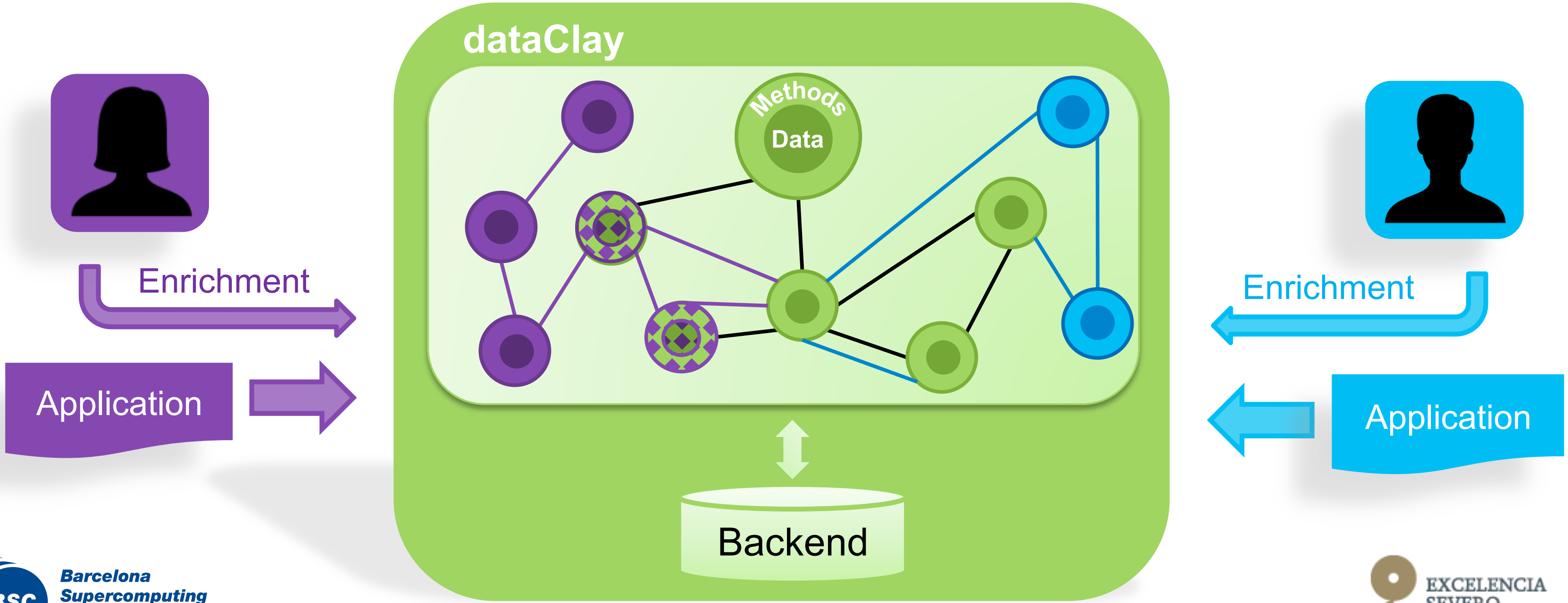


Goal: provide persistent objects infrastructure integrated as naturally as possible with the programming language and with the COMPSs inherent concurrency

dataClay

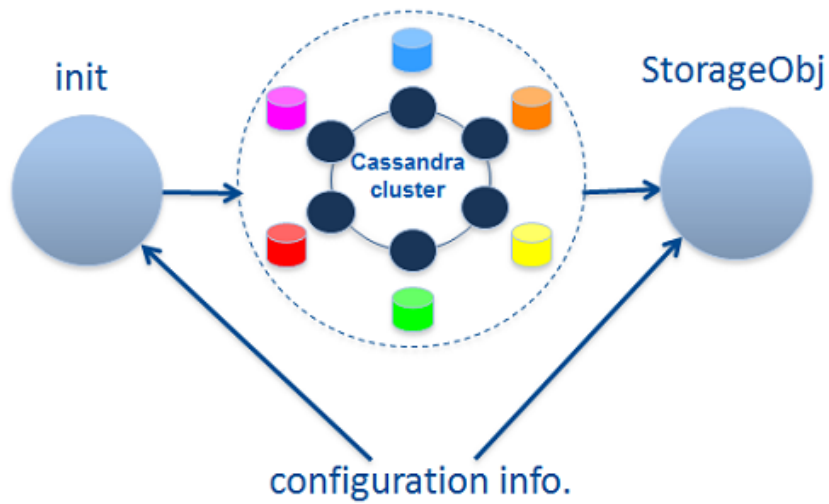
- dataCLay: platform that manages **Self-Contained Objects** (data and code)

- Platform features:
  – Store and retrieve objects as seen by applications
  – Remote execution of methods
  – Add new classes
  – Enrich existing classes: With new methods and with new fields

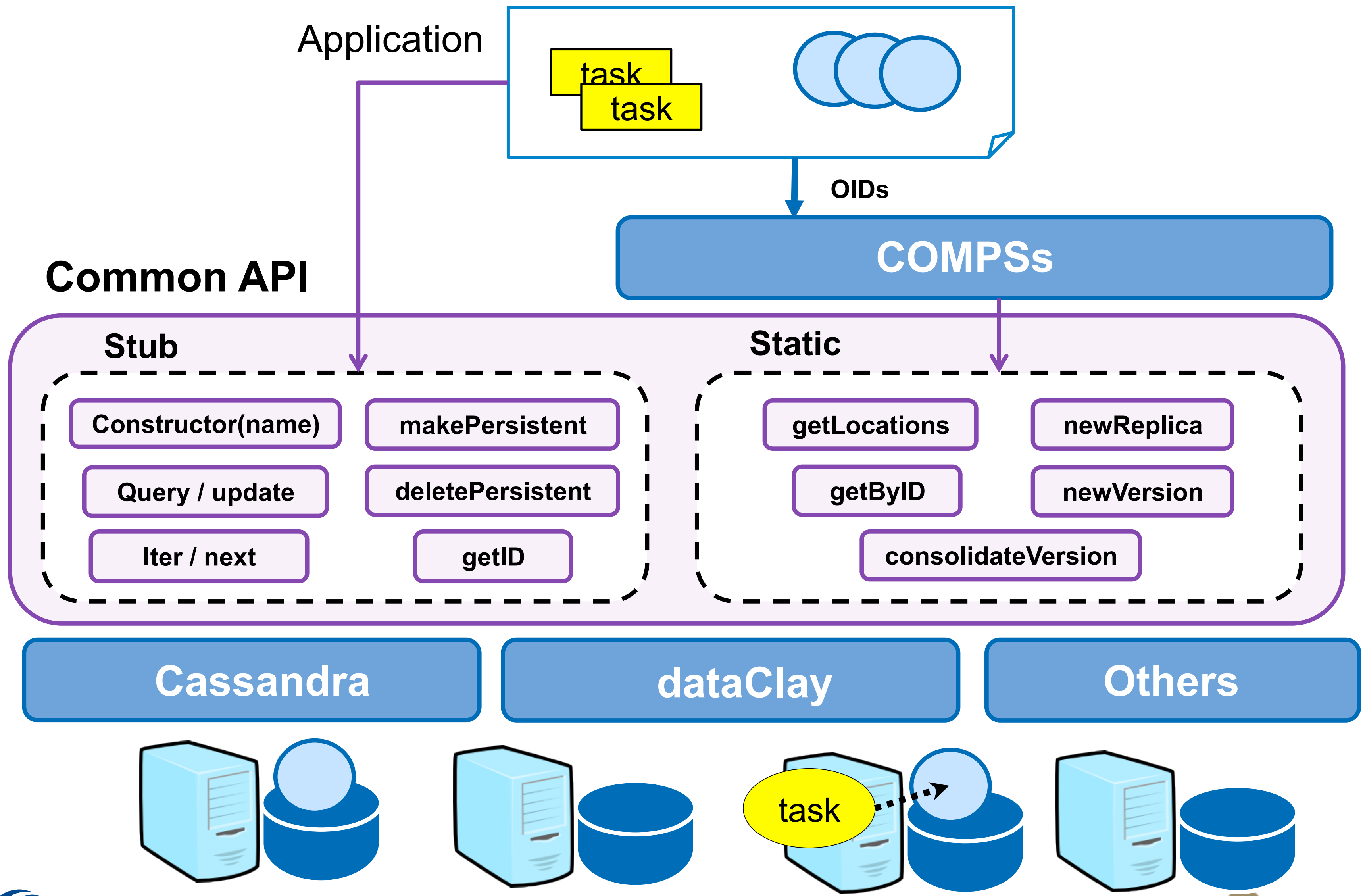


Hecuba

- Set of tools and interfaces that aim to facilitate an efficient and easy interaction with non-relational data-bases
- Currently implemented on Apache Cassandra database
  - However, easy to port to any non-relational key-value data store
- Mapping of Python dictionaries into Cassandra tables
  - Both consist on values indexed by keys
  - Only Python data type supported right now
- Redefinition of Python iterators
  - Accessing blocks of keys
  - Exploiting locality



Integration COMPSs – Common Storage API



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

PyCOMPSs and data persistency

Class definition

```
class Foo(object):
   """ Property bar int """
   def init (self, val):
            self.bar = val
```

- PyCOMPSs objects can be made persistent
- Tasks can operate on persistent and not persistent objects
- PyCOMPSs runtime favours locality by scheduling tasks on the resource where the object is stored

Producer

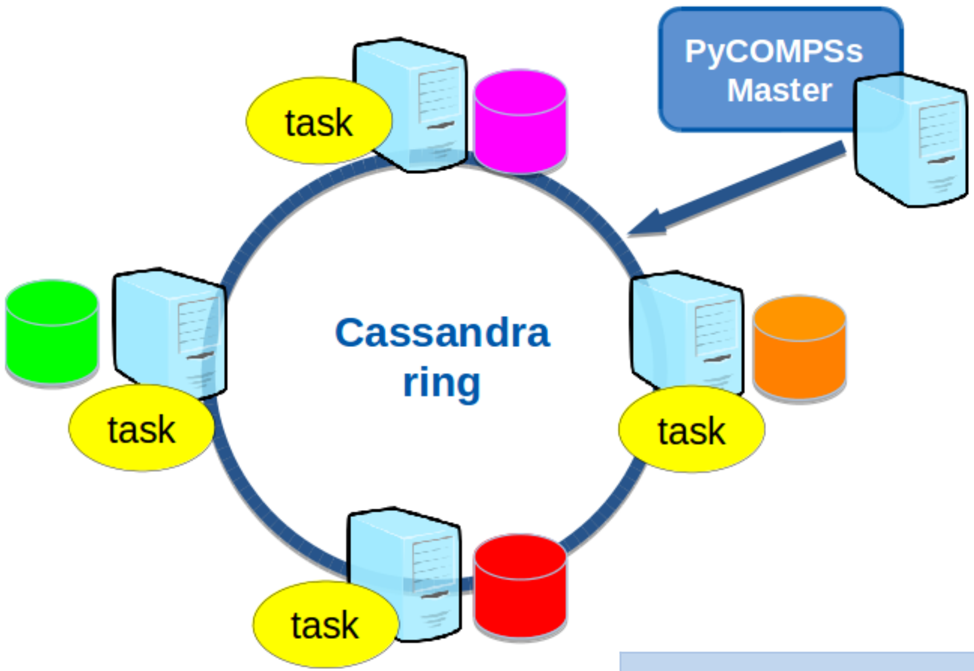
```
@task()
def my func(foo1, foo2):
   sum = foo1.bar + foo2.bar
   print 'Sum:', sum

o1 = Foo(1)
o2 = Foo(2)
...
o1.make persistent('MyFooObject')
...
my func(o1, o2)
```

Consumer

```
@task()
def another func(foo):
...
o = Foo('MyFooObject')
...
another func(o)
```

Use of identifiers

COMPSs + Hecuba: prototype implementation



Minerva cluster
- 5 nodes
- 2 Intel Xeon Quad-Core L5630 2.13GHz, 24 GB RAM
- 6 TB HDD
- Gigabit Ethernet

	Cassandra topology	PyCOMPSs workers	Time (secs)
	4, 4, 0, 0	4, 4, 0, 0	19228
	8, 8, 0, 0	8, 8, 0, 0	10273
hyperthreading	16, 16, 0, 0	16, 16, 0, 0	12867
hyperthreading	16, 16, 16, 16	16, 16, 16, 16	6644

Summary

- Task-based programming is an approach based of sequential programming that is able to deploy scientific workflows
- BSC approach is the StarSs programming model, with different implementations
- COMPSs and its binding to Python (PyCOMPSs) has been designed taking into account the following aspects
  - Programmability
  - Portability
  - Performance optimization
  - Integration with new efficient data access approaches
- Current developments consider the integration with new storage technologies in order to face the BigData challenges

Human Brain Project

- A 10-year European initiative to understand the human brain, enabling advances in neuroscience, medicine and future computing

- One of two FET Flagships

- A consortium of 256 researchers from 146 institutions, in 24 countries across Europe, in the US, Japan and Chin

- BSC contributes with programming models and resource management

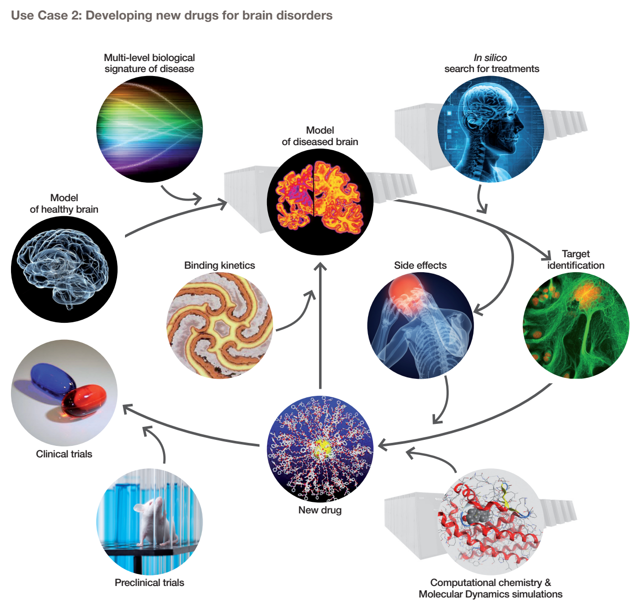


Figure 29: Use of the HBP platforms to accelerate drug development. Optimizing drug discovery and clinical trials

Human Brain Project

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Severo Ochoa

- The BSC-CNS has been accredited with the Severo Ochoa Center of Excellence, an award given by the Spanish Ministry as recognition of leading research centres in Spain that are internationally known organisations in their respective areas.
- Involves all BSC R&D departments
- Four subprojects:
  - Hardware and software technologies, to facilitate the introduction of Exascale computing and managing large amounts of data, focusing on the improvement of energy efficiency
  - Personalized medicine, to design drugs to fit the needs of each patient
  - Heart simulation, to perform modelling and simulation with the primary objective to determine how the heart muscle works
  - Air quality and climate models, specially in areas that affect health (Sahara dust concentration)

EXCELENCIA SEVERO OCHOA

COMPSs

- Project page: http://www.bsc.es/compss
- Direct downloads page: http://www.bsc.es/computer-sciences/grid-computing/comp-superscalar/download
  - Source code
  - Sample applications & development virtual appliances
  - Tutorials
  - Red-Hat & Debian based installation packages

The COMPSs team

- Rosa M Badia
- Pol Alvarez (part time)
- Javi Conejero
- Sandra Corella (part time)
- Carlos Diaz
- Jorge Ejarque
- Fredy Juarez
- Daniele Lezzi
- Francesc Lordan
- Cristian Ramon
- Raul Sirvent

Other CS members

- Toni Cortes
- Anna Queralt
- Jonathan Martí
- Jordi Torres
- Yolanda Becerra
- David Carrera
- Jesus Labarta
- Eduard Ayguadé

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Thank you!

Downloads: **http://www.bsc.es/computer-sciences/grid-computing/comp-superscalar/download**

Support mailing list at **http://compss.bsc.es/support-compss**

Announces mailing list at **http://compss.bsc.es/announces-compss**