www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Easy Programming the Cloud with PyCOMPSs

FiCLOUD 2014
Barcelona, August 28

Human Brain Project

EXCELENCIA SEVERO OCHOA

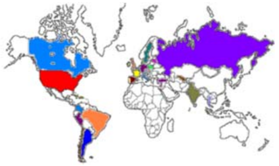
Barcelona Supercomputing Center

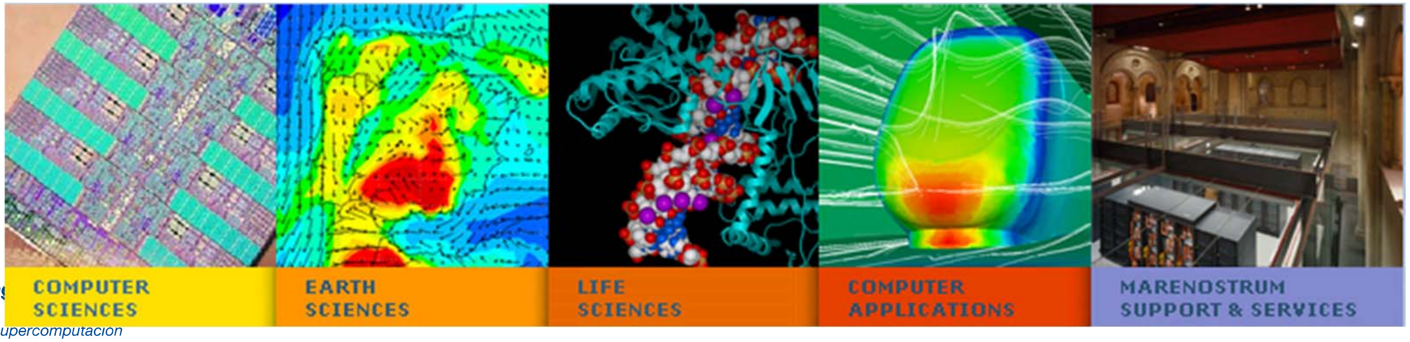
- The BSC-CNS objectives:
  - R&D in Computer Science ences and Earth Sciences
  - Supercomputing support ernal research
- BSC-CNS is a consortium that includes :
  - the Spanish Government – 51%
  - the Catalonian Government – 37%
  - the Technical University of Catalonia (UPC) – 12%
- +400 people

GOBIERNO DE ESPAÑA MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD

Generalitat de Catalunya
Departament d'Economia i Coneixement

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH



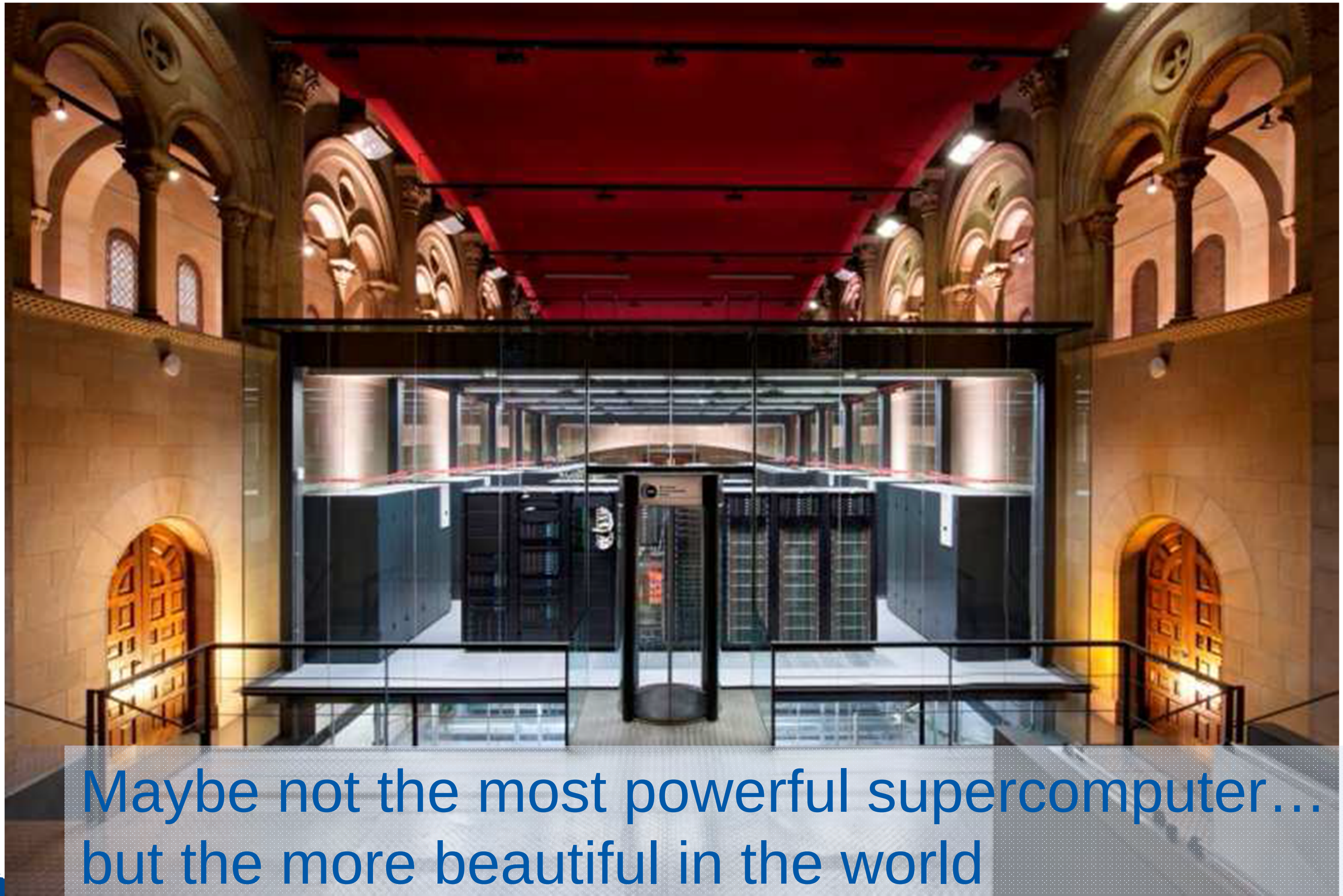


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

MareNostrum III overview

- Deployed end on 2013
- 36 x IBM iDataPlex Compute racks
  - 84 x IBM dx360 M4 compute nodes
    - 2x SandyBridge-EP E5-2670 2.6GHz/1600 20M 8-core 115W
    - 8x 4G DDR3-1600 DIMMs (2GB/core)
    - 500GB 7200 rpm SATA II local HDD
- 3028 compute nodes
  - 48,448 Intel cores
- Memory 94.62 TB
  - 32GB/node
- Peak performance: 1.0 Pflop/s
  - Node performance: 332.8 Gflops
  - Rack Performance: 27.95 Tflops
  - Rack Consumption: 28.04 kW/rack (nominal under HPL)
- Estimated power consumption: 1.08 MW
- Infiniband FDR10 non-blocking Fat Tree network topology
- Position 41 of TOP500

MareNostrum III

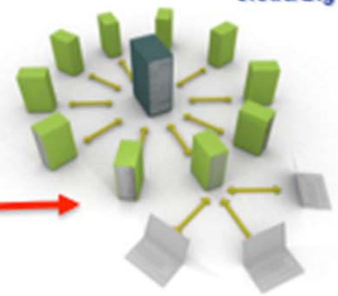


Maybe not the most powerful supercomputer…
but the more beautiful in the world

BSC Supercomputing Center
Centro Nacional de Supercomputación

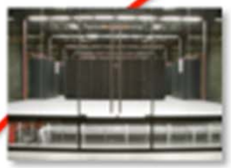
BSC-CNS: Computer Sciencies

Cloud/Big Data



Programming models:
- StarSs programming model
- Accelerators (CUDA, OpenCL)
- Influencing standards (OpenMP, OpenACC)
- Data movement- and power-conscious scheduling
- Hybrid MPI/task parallelism, dynamic load balancing
- Transactional memory and resilience
- DSLs

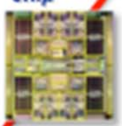
System design:
- Low-power supercomputing (PRACE prototype and Montblanc project)



Future Exaflop systems

Distributed and Cloud computing:
- Programming models: COMPSs, MapReduce
- Resource management: virtualization in data centers, performance and power-aware job and application scheduling
- Efficient and reliable resource management for Big Data applications
- Parallel file system scalability and I/O for Cloud

Multicore chip



Performance analysis and prediction tools:
- Tracing scalability
- Pattern and structure identification
- Visualization and analysis
- Processor, memory, network
- Node and microarchitecture level simulators (TaskSim)

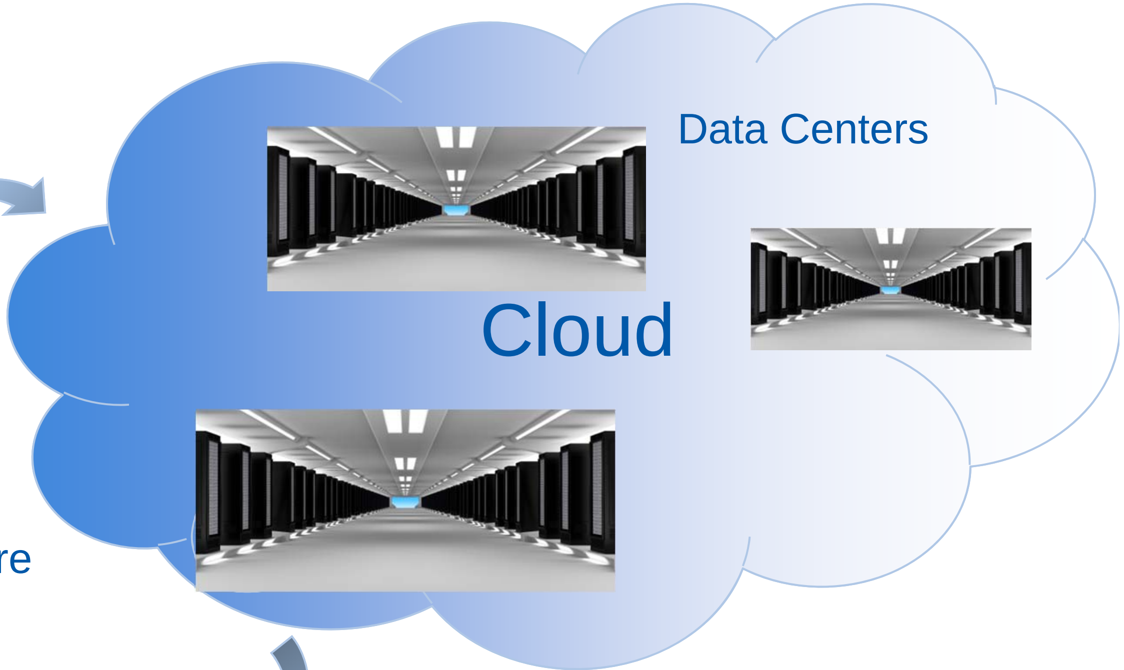
Multicore architectures:
- Massive multithreaded architectures
- Low-power vector architectures for media
- Architectures for real-time
- Architecture support for programming models and runtimes
- Interconnection networks

Outline

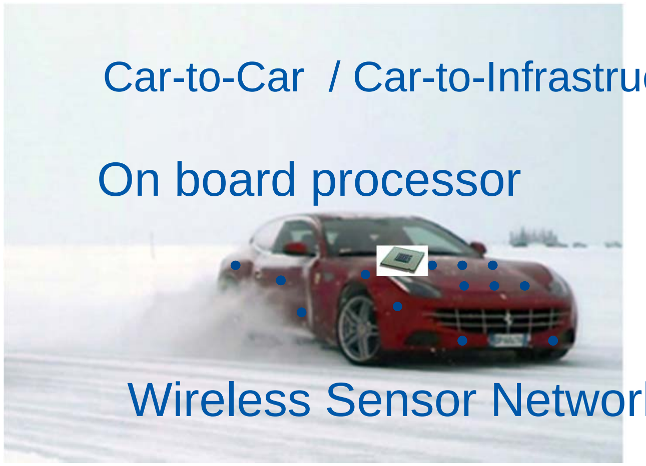
- Programming challenges for the cloud and distributed computing in general
- StarSs programming model
- Programming the cloud with COMPSs
  - Syntax + Python binding: PyCOMPSs + C
  - COMPSs infrastructure and features
  - Associated Tools: IDE, monitor, traces
- Integration of COMPSs with new storage strategies
- Other projects where COMPSs has been involved
- Conclusions

Challenges: How to efficiently compute in the cloud with wireless sensor networks data?

Active aerodynamics simulation



Reliable, secure, ubiquitous communication network
CAR STATE COMMUNICATION

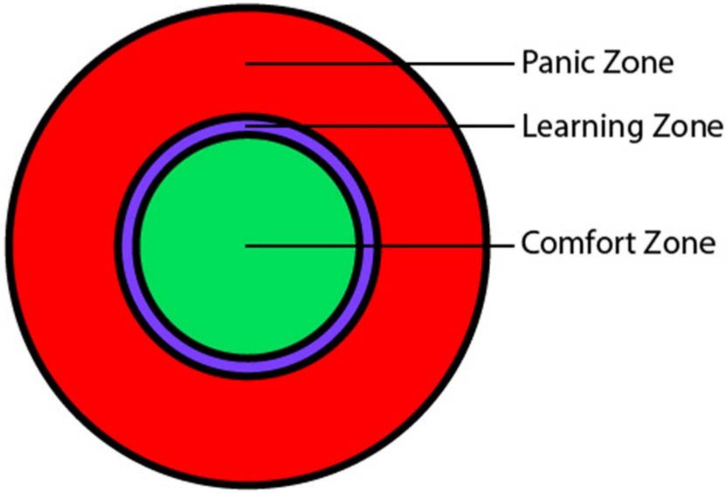


Supercar

Real-time control of single vehicles
ACTION on the ACTUATORS
in cooperation with the on board processor

Cloud programming challenge, or how to make it the programmers comfort zone

Social pedagogy



* The Learning Zone Model (Senninger, 2000)

- The Learning Zone model establishes a theory of how performance of a person can be enhanced and their skills optimized
  - Comfort Zone: feel comfortable and do not have to take any risks
  - Learning Zone: just outside of our secure environment, we grow and learn
  - Panic Zone: all our energy is used up for managing/controlling our anxiety and no energy can flow into learning.
- Moving to the learning zone, enables to extend the comfort zone, moving towards the panic zone
- When following a personal dream or vision, individuals need to move to the learning zone and take controlled risks, in order to achieve the challenges of their panic zone

Cloud poses different challenges to programmers
… away from the current comfort zone
… maybe in the panic zone???

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

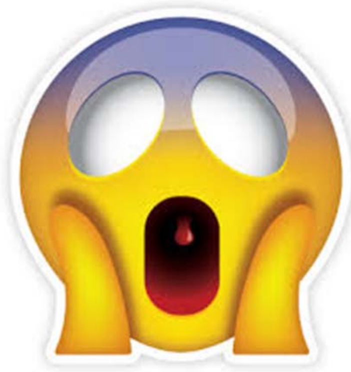
The programming comfort zone

- State of the art in programming
  - Sequential programming
  - Data is always where you expect
  - All decisions controlled by the programmer

Comfort zone

- Programming for the cloud
  - Parallel programming (~)
  - Elasticity
  - Distributed environment -> where is my data?

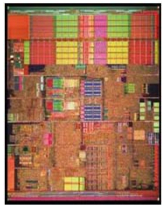
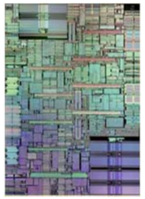
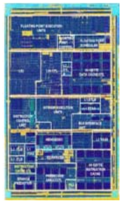
Panic? Zone



Sequential programming

Applications

Programming language

Programs “decoupled” from computing platform

Simple interface
Sequential program

Regular processors

Programming evolution for distributed programming

- Distributed computing APIs make programming more complicated

Applications

Programming language + API

Program logic
+
Middleware specificities

BSC vision

Applications

PM: High-level, clean, abstract interface

Power to the runtime

API

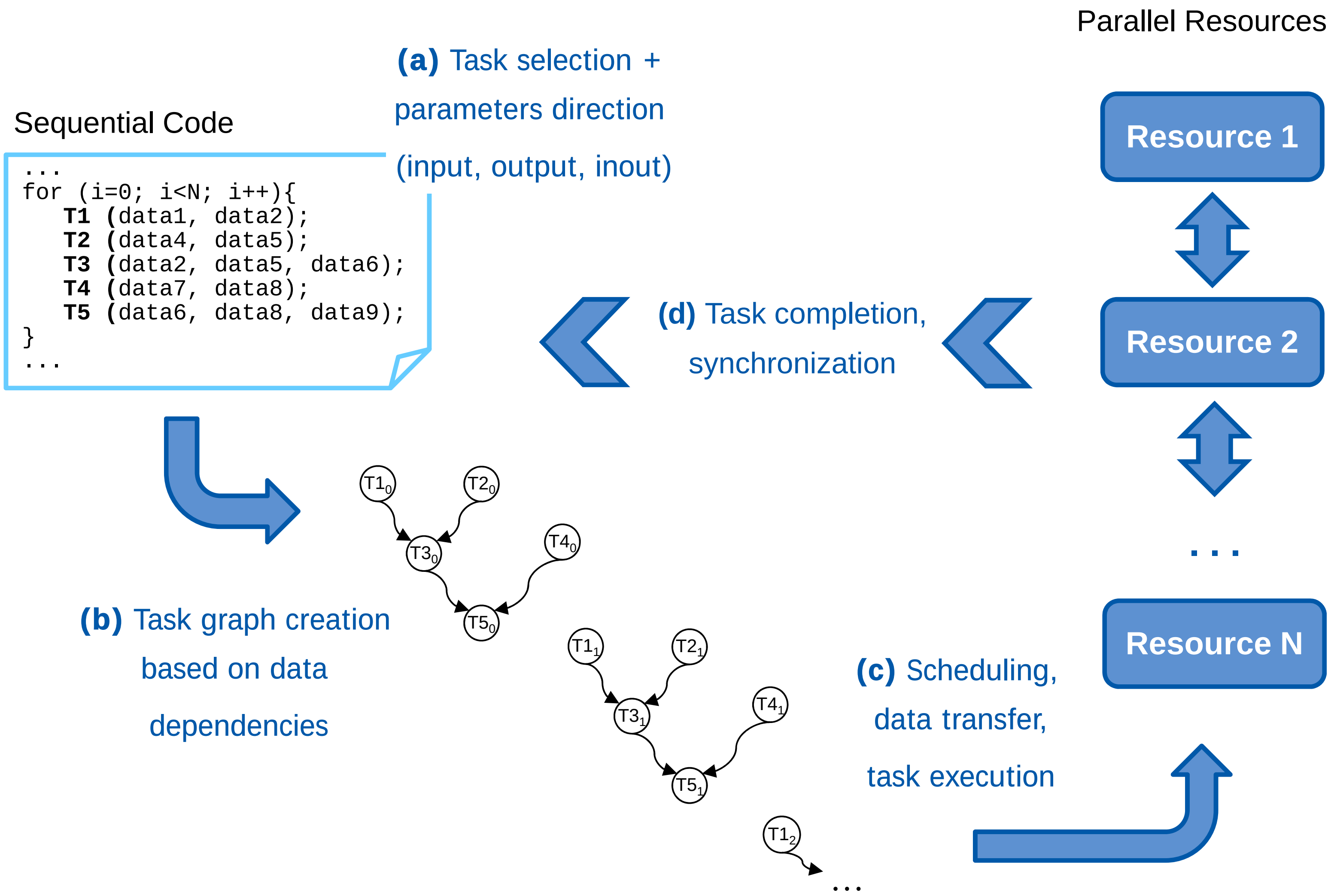
Program logic independent of computing platform

General purpose
Task based
Single address space

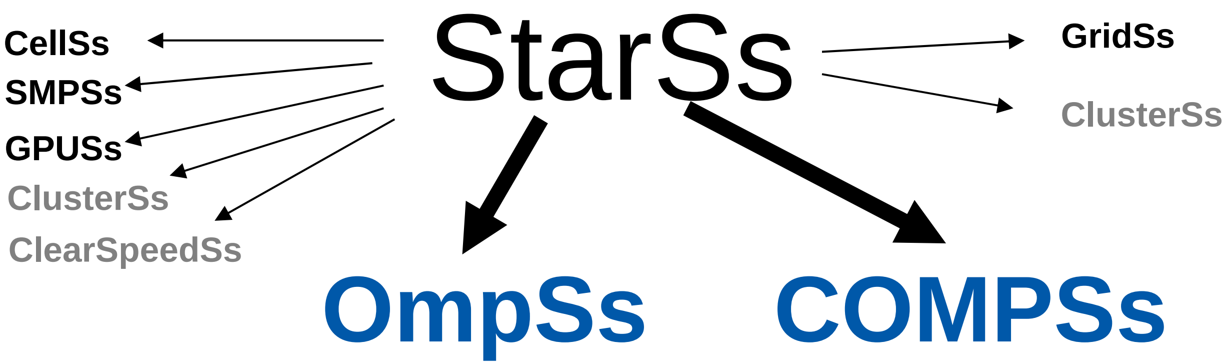
“Reuse” architectural ideas under new constraints

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

STARSs basic idea



The StarSs programming model



- StarSs
  - Sequential general purpose programming language + annotations
  - Task based
  - Simple linear address space
  - Support for SMP, GPUs, Cluster, Grids and Clouds

Open Source
http://compss.bsc.es

- Programmability/Portability
  - "Same" source code runs on "any" platform
  - Incremental parallelization/restructure
  - Focus in the problem, not in the hardware platform
- Performance
  - Intelligent Runtime
    - Automatically extracts and exploits parallelism
    - Locality awareness
    - Matches computations to specific resources on each type of target platform

The StarSs "Granularities"

StarSs

OmpSs	**COMPSs**
@ SMP *@ GPU* *@ Cluster*	
Average task Granularity:	
100 microseconds – 10 milliseconds	1second - 1 day
Address space to compute dependences:	
Memory	Files, Objects (SCM)
Language binding:	
C, C++, FORTRAN	Java, Python, C/C++

Parallel ← → **Ensemble, workflow**

BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Advantages and drawbacks of COMPSs

- ✔ More flexible and with more expressivity
  - The potential of the programming language
  - Enables to express complex problems
- ✔ Data independent
  - Different data inputs may generate different task graphs
- ✔ Powerful runtime
  - Platform unaware
  - Exploits inherent parallelism
- ✗ Less explicit than graphical workflows
  - Although this can be partially compensated with the COMPSs monitor
- ✗ Large degree of flexibility may prevent some programmers to be efficient
  - Schemas such as MapReduce are sometimes more appreciated by programmers
  - Can be improved through training and support

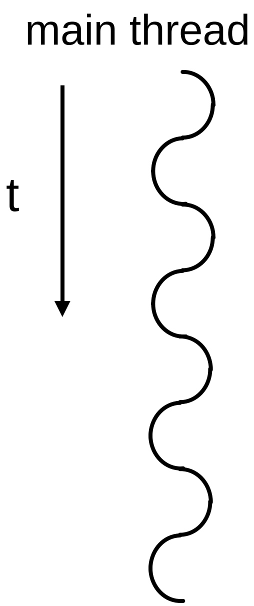
Programming objectives

- Reduce the development complexity of Grid/Cluster/Cloud applications to the minimum
  – Writing an application for a computational distributed infrastructure may be as easy as writing a sequential application

- Target applications: composed of tasks, called several times
  – Granularity of the tasks or programs
  – Data: files, objects, arrays and primitive types
- Programming languages support
  – Java (native)
  – Python
  – C/C++

COMPSs syntax: Java

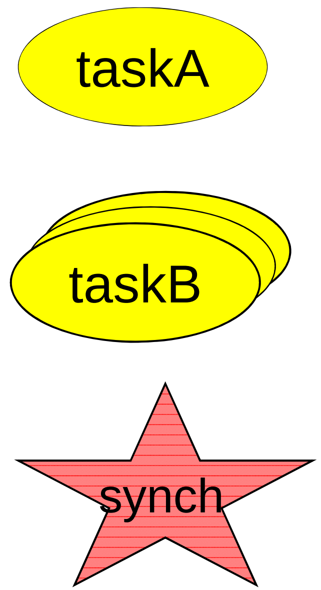
- Based on pure-Java fully-sequential programming
  - No APIs, no threading, no messaging
  - No parallel constructs, no pragmas
  - Maintains sequential consistency




```
Main Program {
  taskA(data1);

  for (int i=0; i< N; i++)
    taskB(data1, data2);

  if (condition)
    process(data2);
}
```



COMPSs syntax: Java

Annotated Interface

```
public interface HMMPfamItf {
@Constraints(storageElemSize = 0.5f)
@Method(declaringClass = "hmmerws.HMMPfamImpl")
String hmmpfam(
    @Parameter(type = Type.FILE, direction = Direction.IN) String seqFile,
    @Parameter(type = Type.FILE, direction = Direction.IN) String dbFile  );
@Service(namespace = "http://hmmerobj.worker", name = "HmmerObjects", port =
    "HmmerObjectsPort")
String scoreRating(
    @Parameter(type = Type.OBJECT, direction = Direction.IN) String resultFile1,
    @Parameter(type = Type.OBJECT, direction = Direction.IN) String resultFile2  );
}
```

Task constraints

Regular methods

Web services

Parameter metadata

Java code

```
public static void main(String args[]) throws Exception {
        split(fSeq, fDB, seqFrags, dbFrags);
        for (String dbFrag : dbFrags)
            for (String seqFrag : seqFrags) {
                output = HMMPfamImpl.hmmpfam (seqFrag,  dbFrag);
                finalOutput = scoreRating(output, finalOutput);
            }
}
```

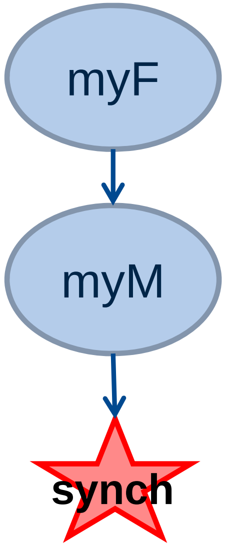
Python (PyCOMPSs) syntax

- Invoke tasks as Python functions/methods
- API for data synchronization

Main Program

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

- Task selection in function definition (decorators)



Function definition

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

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

COMPSs syntax: C

- IDL file used to identify tasks and parameters
- API for data synchronization

Main Program

```
compss_on();

A = Matrix::init(N,M,val);
B = Matrix::init(N,M,val)
C = Matrix::init(N,M,0.0)

C.multiply(A, B);
compss_off();
```

Interface

```
interface Matmul
{
//C functions
void initMatrix(inout Matrix matrix, in int mSize, in int nSize, in
double val);
void multiplyBlocks(inout Block block1, inout Block block2, inout
Block block3);
//C++ Methods
void Block::multiply(in Block block1, in Block block2);
static Matrix Matrix::init(in int mSize, in int bSize, in double
val);
};
```

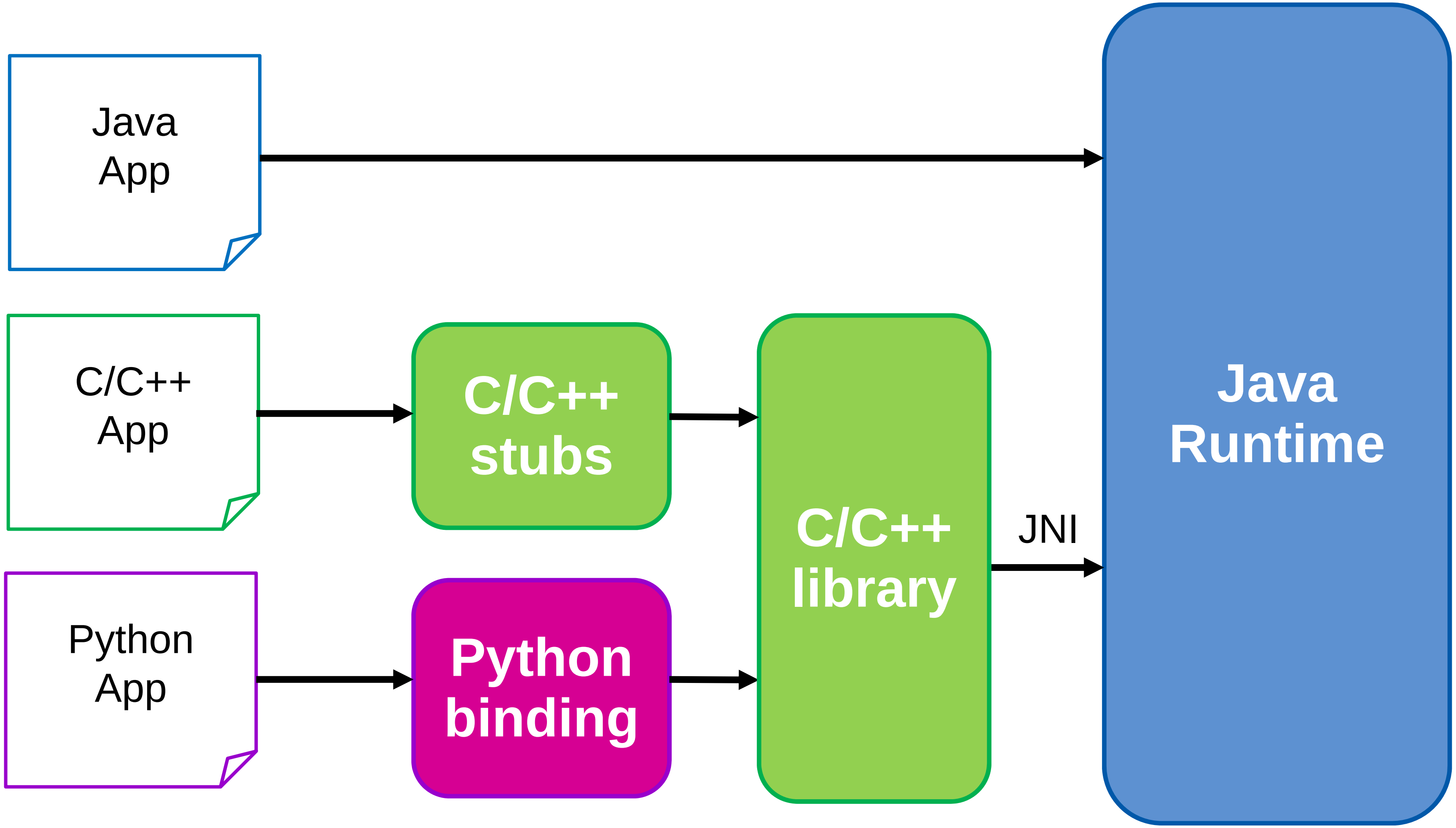
Service Composition

- Orchestrating parallel services as sequential applications
  - Rely on inner services (and methods)
  - Can be published as a service as well
  - Several orchestrations as a single service

```
public class TravelService {
   @Orchestration
   public Booking bookTravel(…) {
      Card c = checkCreditCard(…);
      …
   }
}
```

```
public interface TravelItf {
   @Service(…)
   Card checkCreditCard(…);
   …
}
```

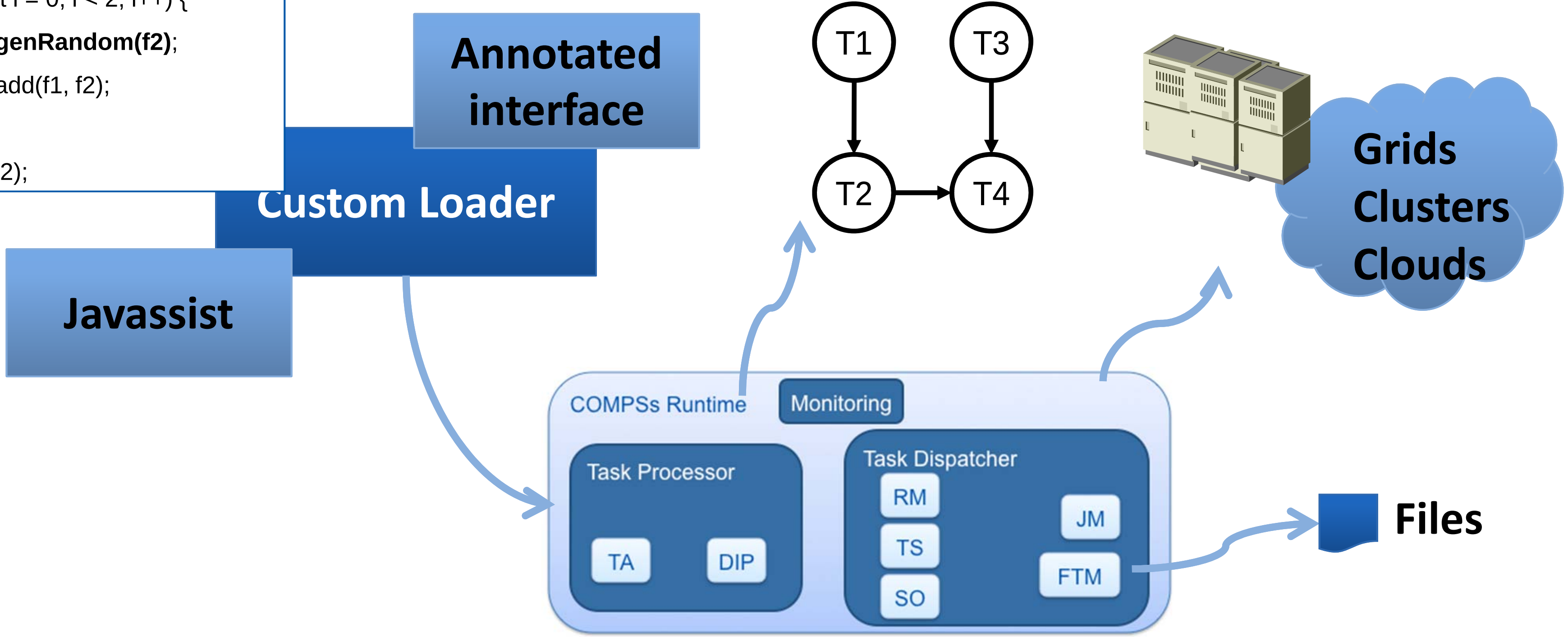
COMPSs Bindings integration



COMPSs Infrastructure

User code: Python, Java, C/C++

```
initialize(f1);
for (int i = 0; i < 2; i++) {
        genRandom(f2);
        add(f1, f2);
}
print(f2);
```



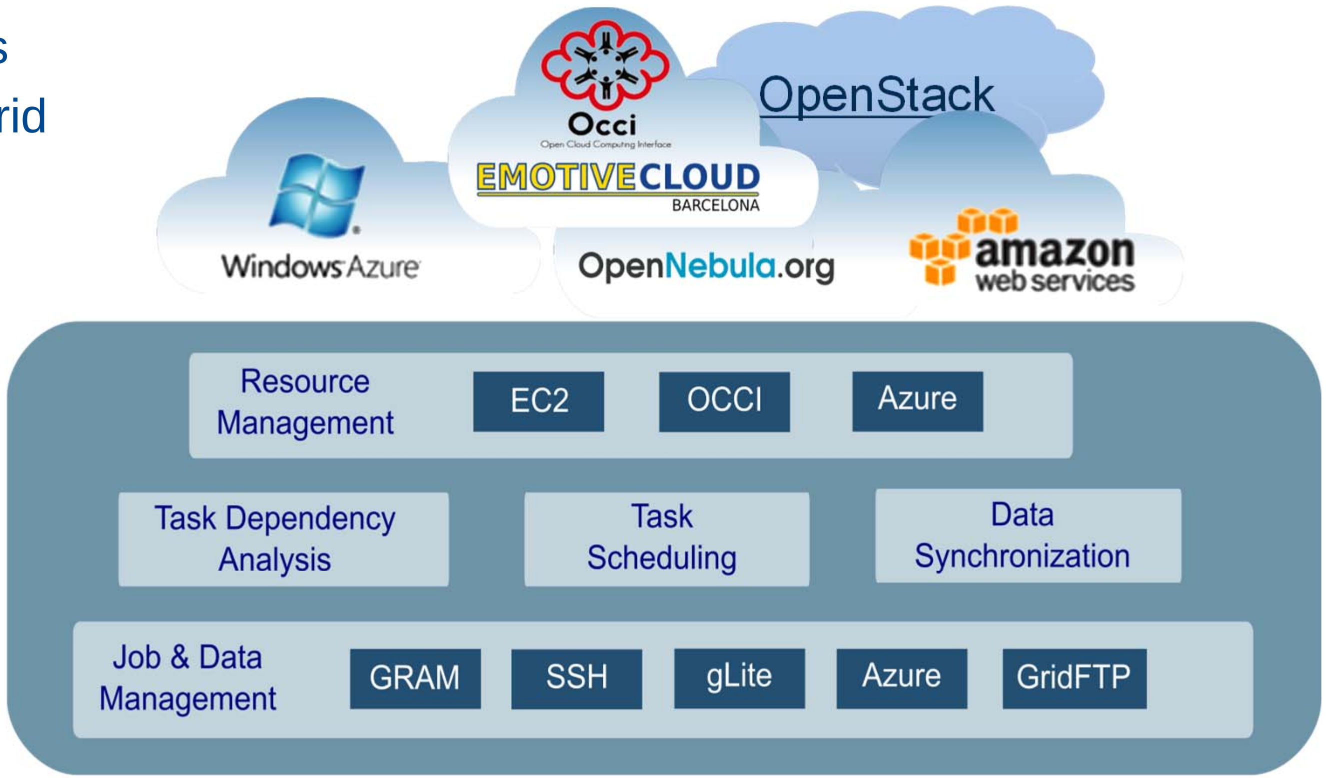
Runtime features

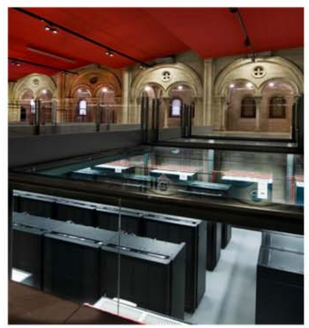
- Supported Features:
  - Unaware of underlying computing platform
  - Interoperability with different clouds
  - Data dependency analysis
  - Data renaming
  - Data transfer
  - Task scheduling
  - Resource management
  - Results collection
  - Fault tolerance
  - Shared disks management

- In Progress:
  - Checkpointing
  - Task nesting
  - Distributed Scheduling P2P
  - Support for heterogeneous platforms
  - Deployment in mobile clouds
  - Task scheduling with multiple versions

Runtime System: Interoperability

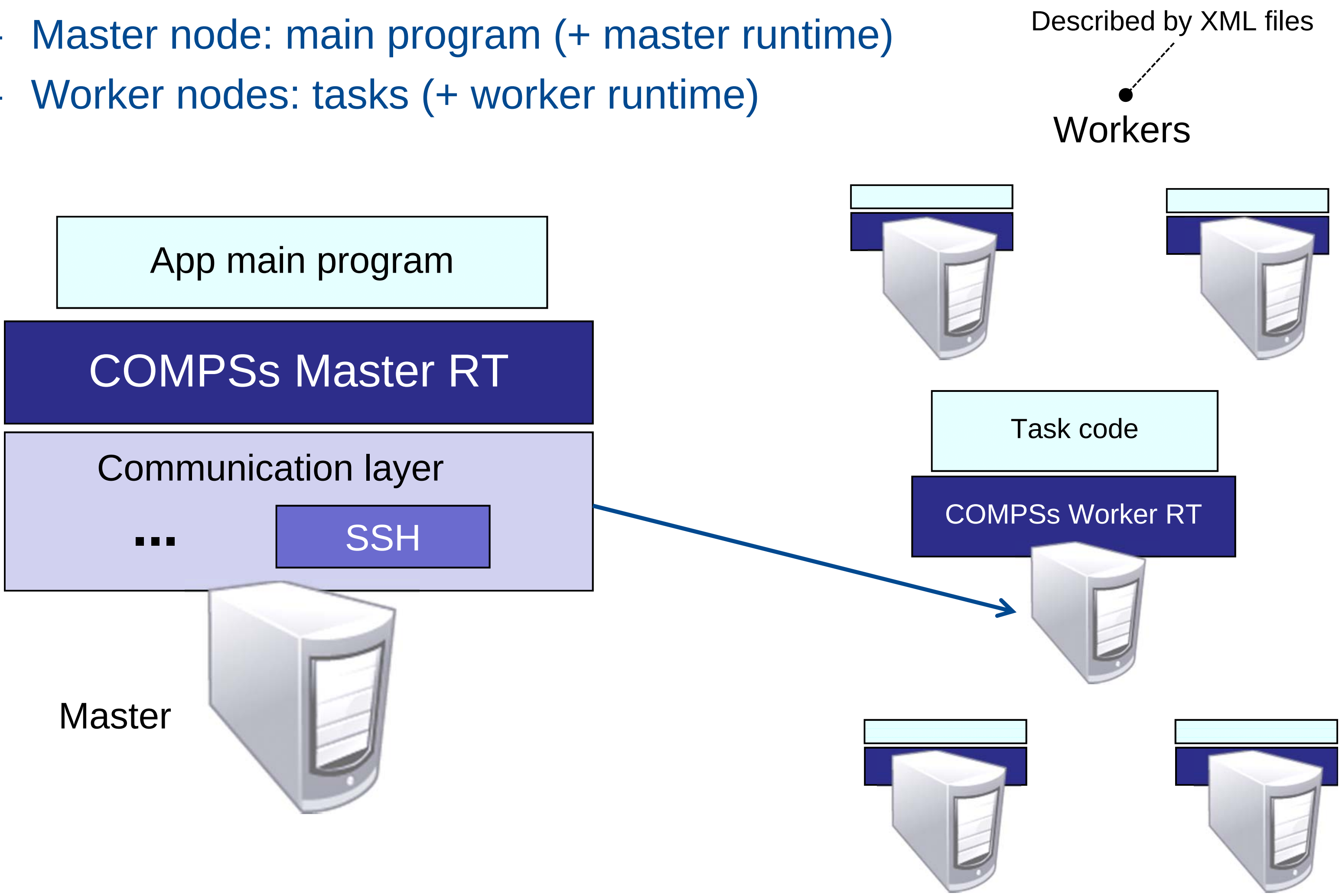
- Platform unawareness
- Support for different grid middlewares
- Cloud interoperability:
  - Public and private
  - Heterogeneous clouds





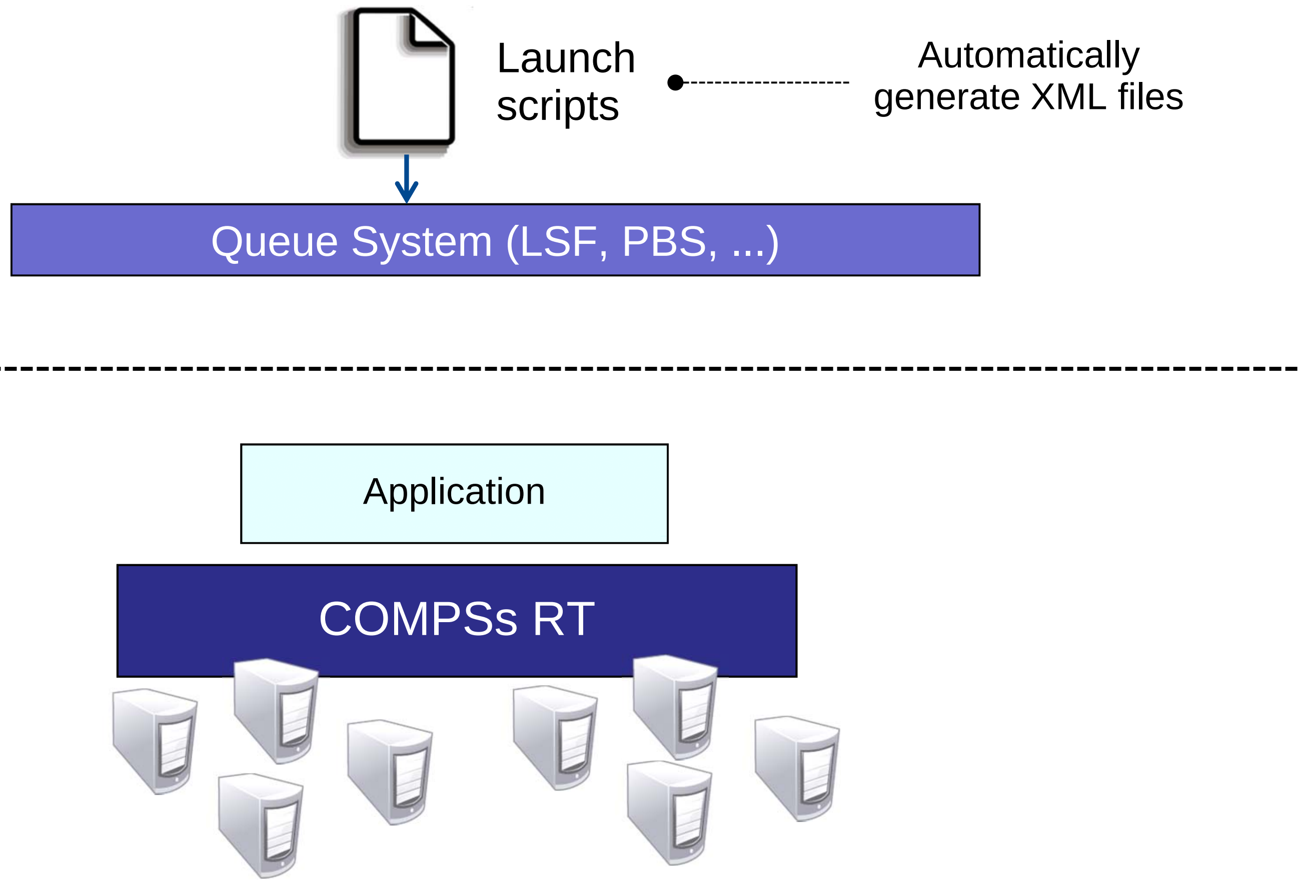
Computing platform: in a Cluster (interactive)

- Typical setup:
  - Master node: main program (+ master runtime)
  - Worker nodes: tasks (+ worker runtime)

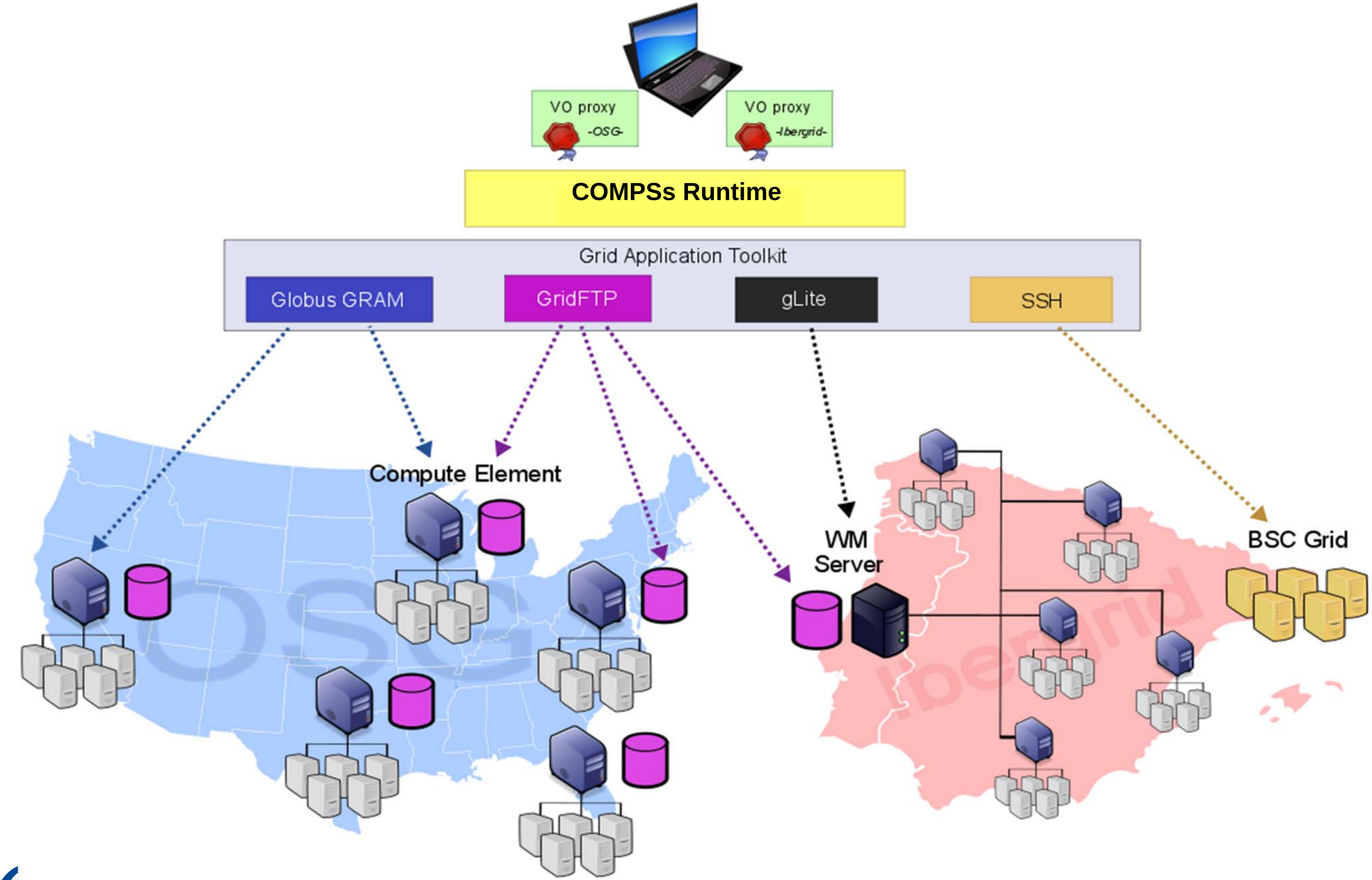


Computing platform: in a Cluster (queue system)

- Execution divided in two phases
  - Launch scripts queue a whole COMPSs app execution
  - Actual execution starts when reservation is obtained



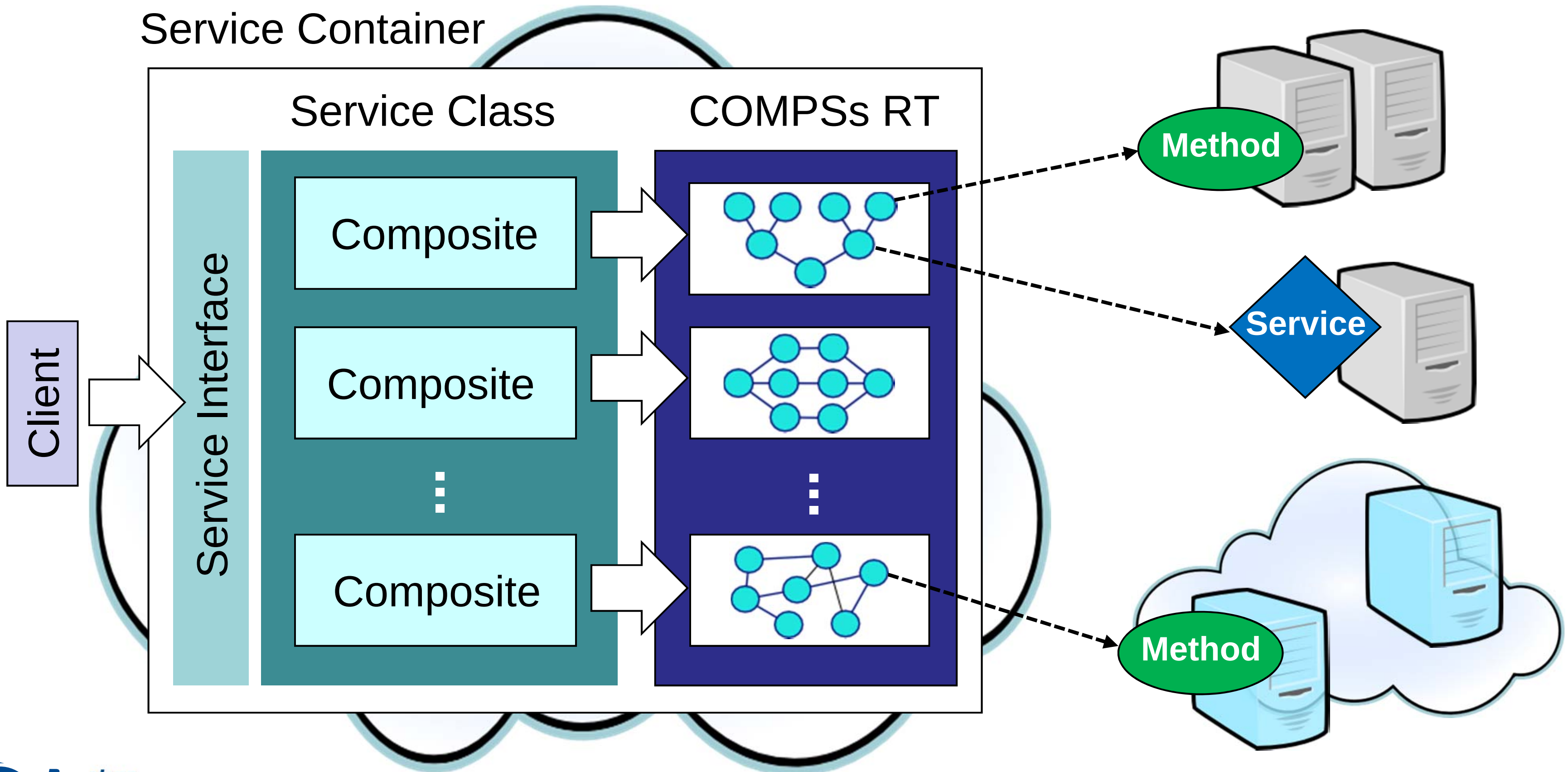
Computing platform: in a Grid



Supercomputing Center
Centro Nacional de Supercomputación

Computing platform: in the Cloud

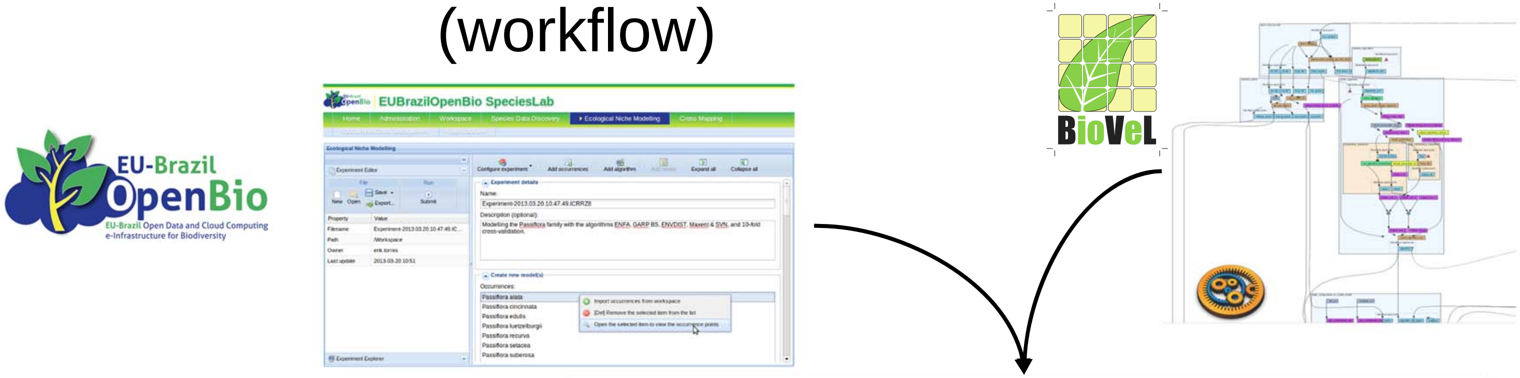
- Runtime integrated in a platform with:
  – Service orientation
  – Virtualization



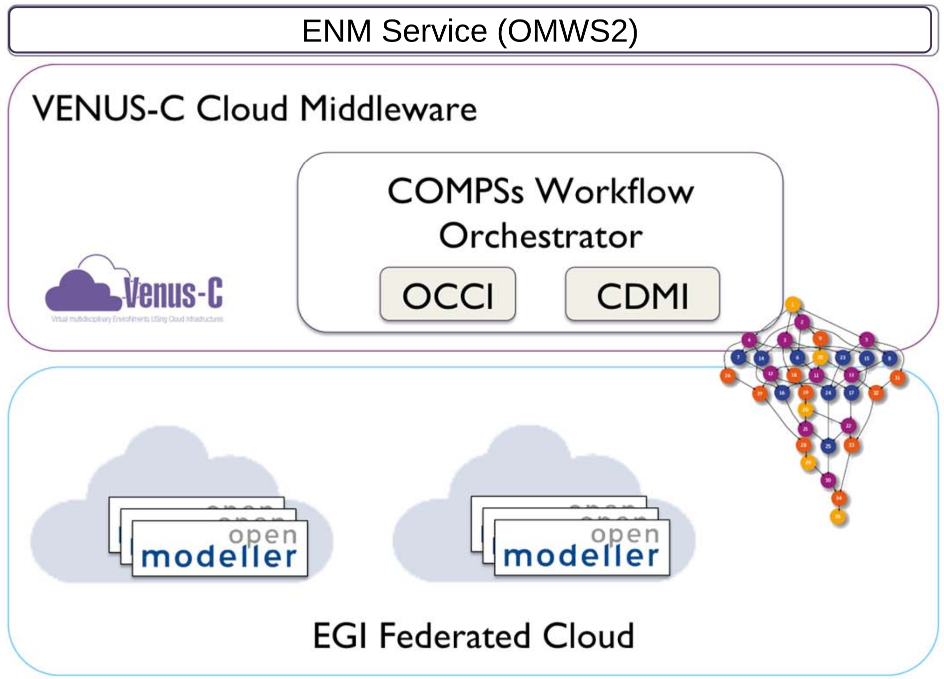
Deployment of COMPSs in EGI federated cloud

Single multi-job request (workflow)

Multiple requests

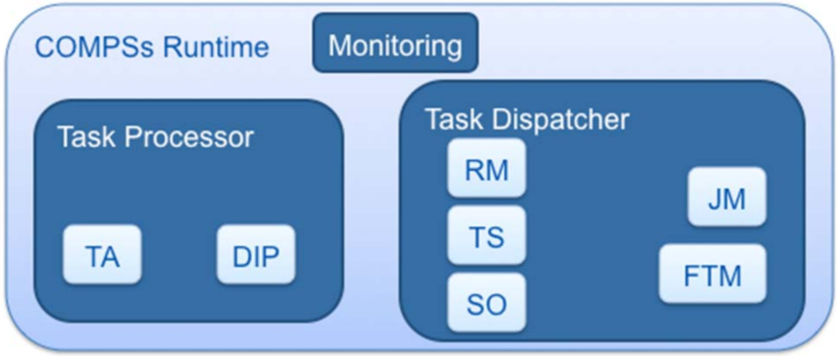


- Single request scenario: Issues a multi-job request (6 species and 2 modelling algorithms) producing 12 distribution models. It exploits different FedCloud templates.
- Multiple requests scenario: tests the global service performance for a given workload pattern (Gaussian random) by issuing many requests with low complexity (3 tasks).



Runtime behavior: scheduling and resource management

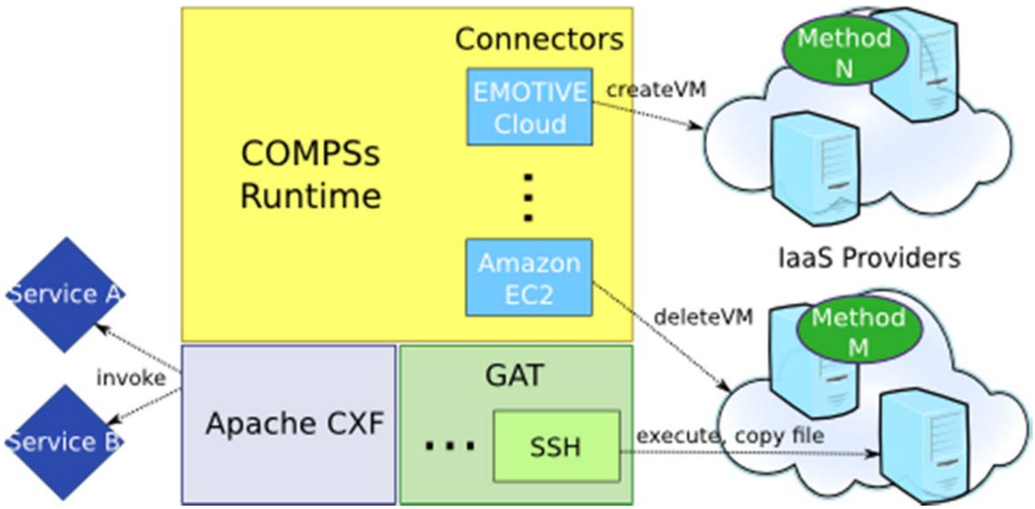
- Task Scheduler
  - Assigns tasks to VMs or physical resources
  - Each VM or resource has its own task queue
- Scheduling Optimizer
  - Checks status of workers
  - Can decide
    - To perform load balancing
    - Create/destroy new VMs



- 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

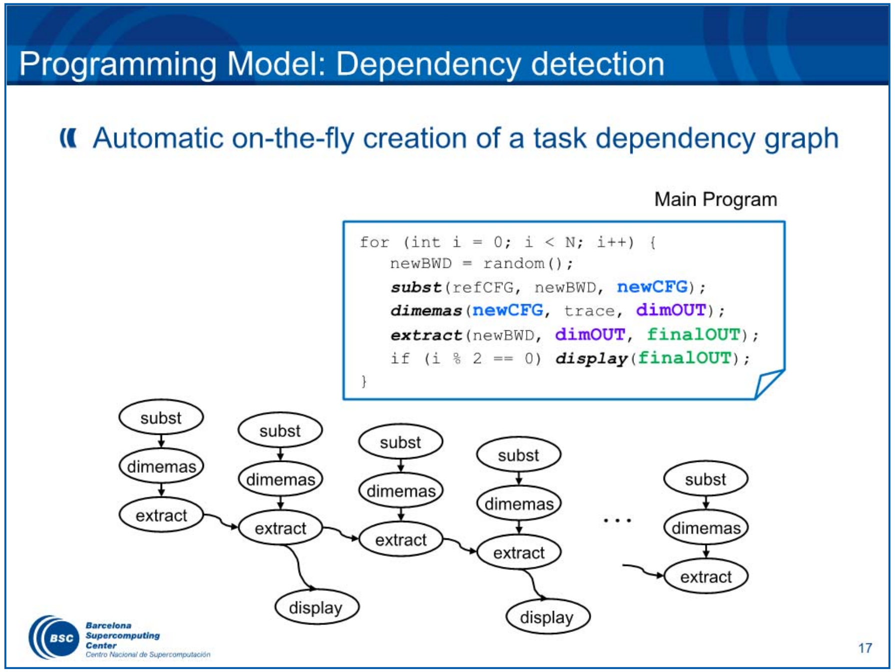
Interoperability to cloud middleware through connectors

- The runtime communicates with the Cloud by means of Cloud connectors
- The connectors implement the interaction of the runtime with a given Cloud provider
- Connectors abstract the runtime from the particular API of each provider
- This design facilitates the addition of new connectors for other providers.



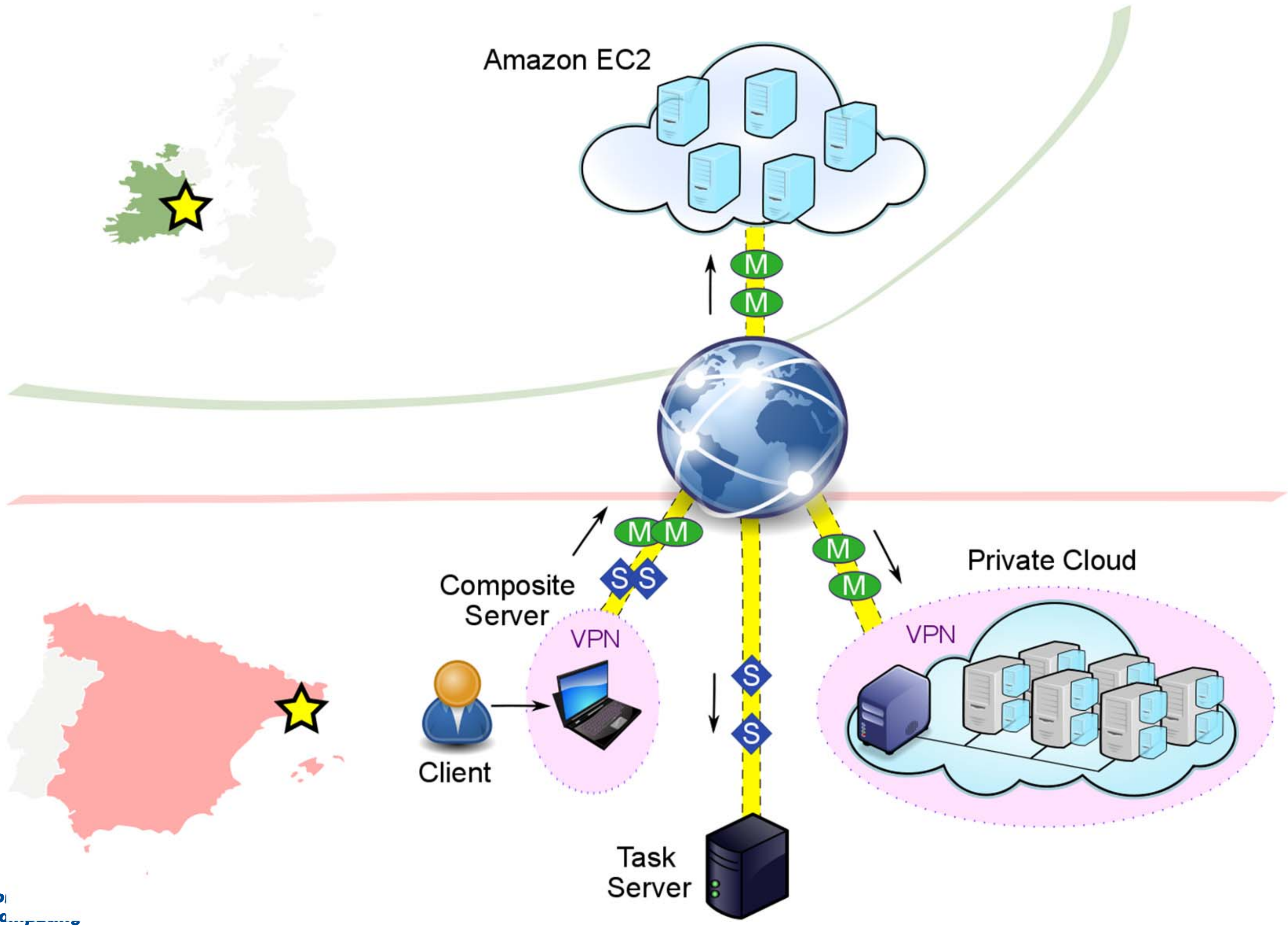
Runtime behavior

- Dependence detection
  - In files
  - In objects
  - In data from Web services

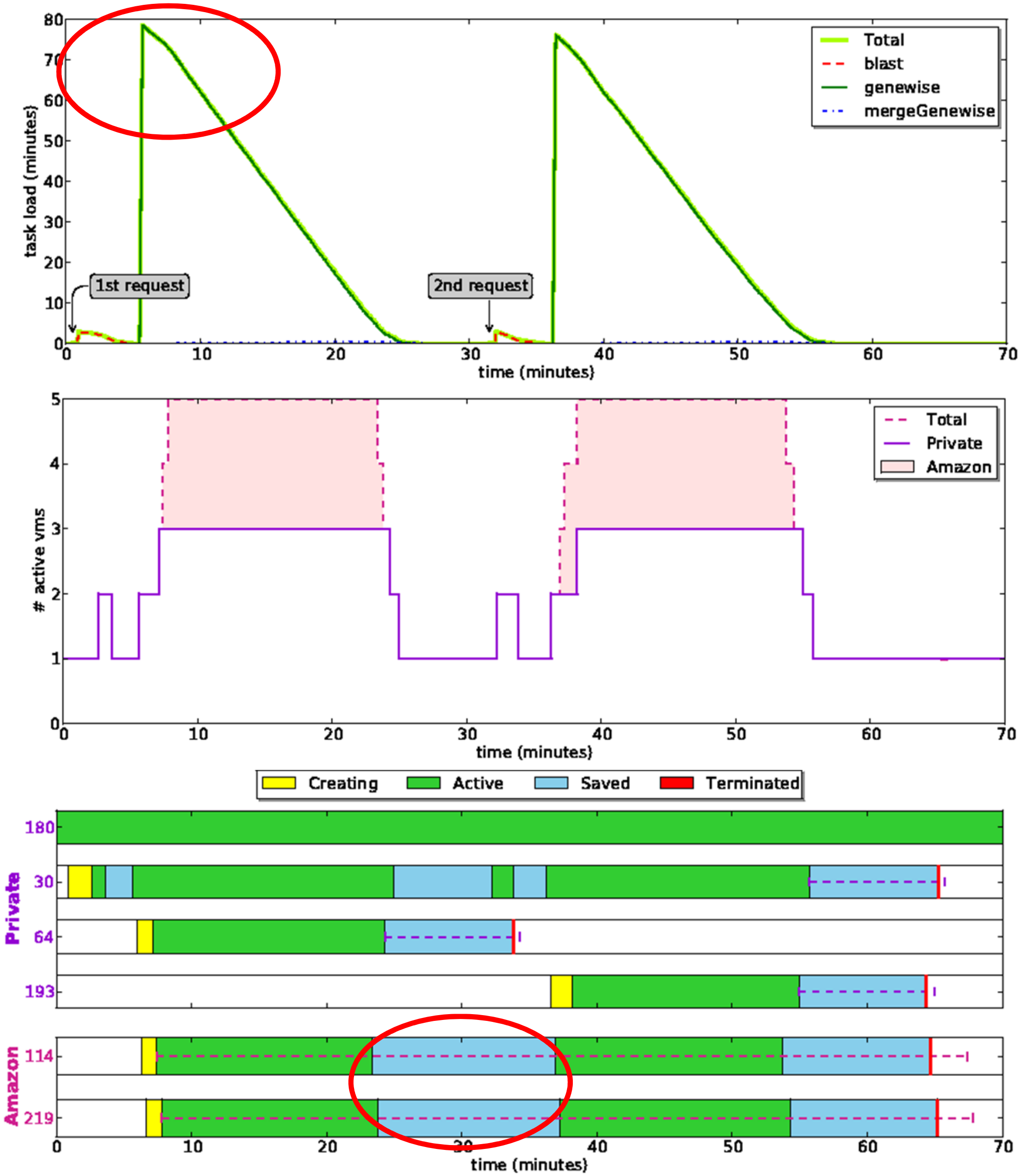


Elasticity in the Cloud

- Sample hybrid setup for Cloud bursting



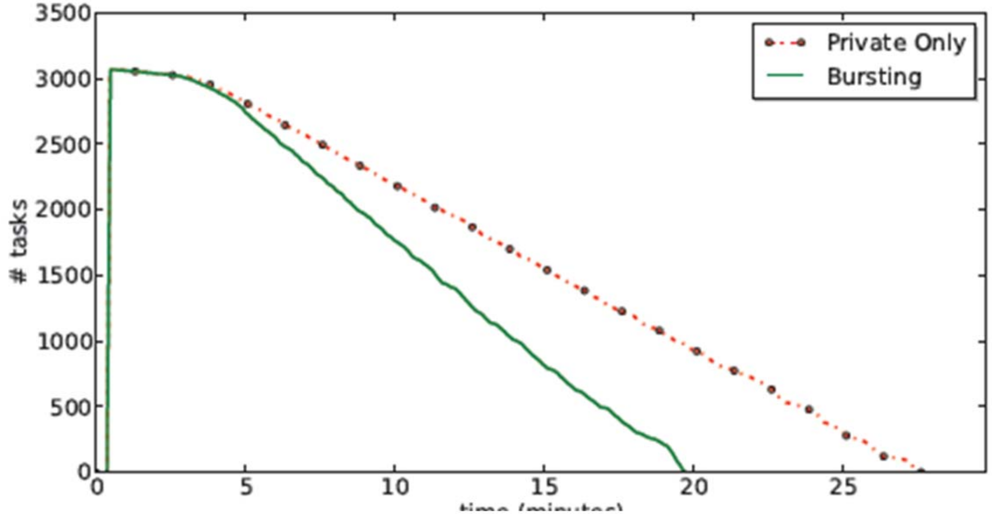
Elasticity in the Cloud

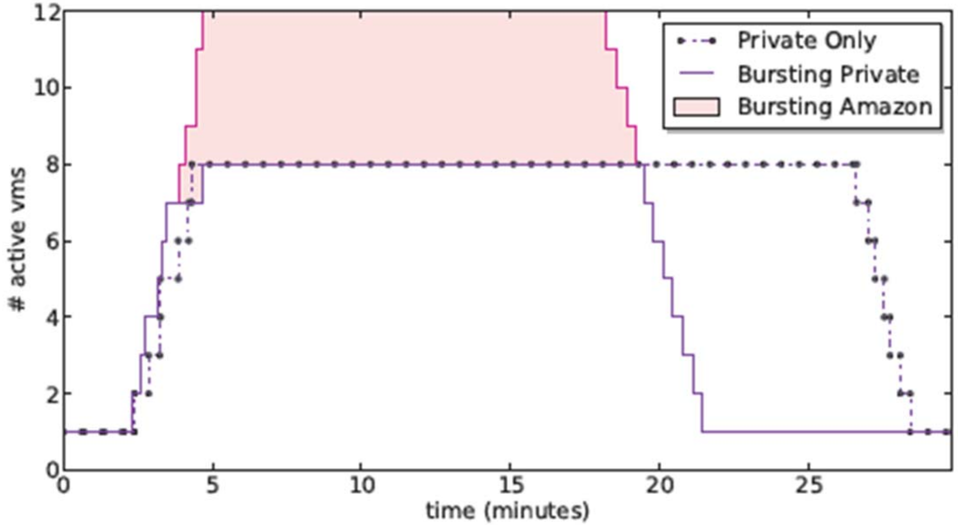


- Dynamic creation / destruction of VMs
  - Depending on task load
- Bursting to meet peak demands
  - Private Cloud (EMOTIVE)
  - Public Cloud (Amazon)
- Save VMs for later use
  - Amazon: use the whole hour slot
- Reuse of VMs
- VM deadlines

Elasticity in the Cloud

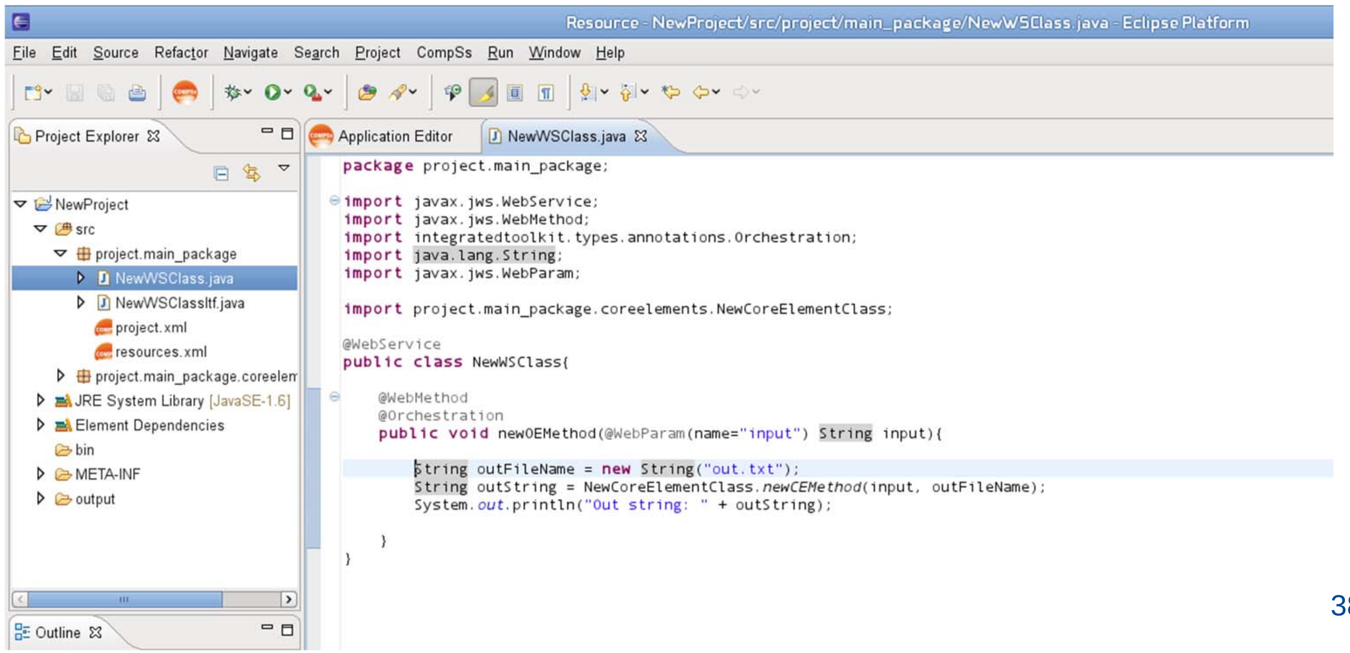
- Scalability
  - Private Cloud: the entire workflow in a single provider
  - Hybrid (Private + Public): tasks and data distributed over two distant providers





COMPSs IDE

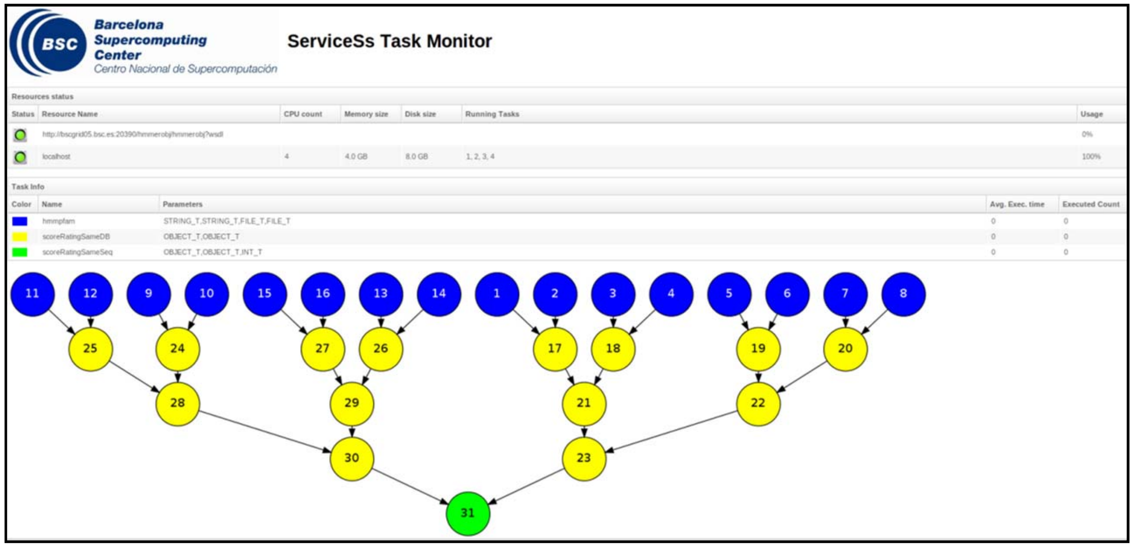
- Graphical interface to help developers with COMPSs applications
  – Annotation of main program and tasks
  – Generation of project and resources files (xml)
  – Deployment in the infrastructure
- Developed as Eclipse plugin
  – Available in the Eclipse marketplace



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

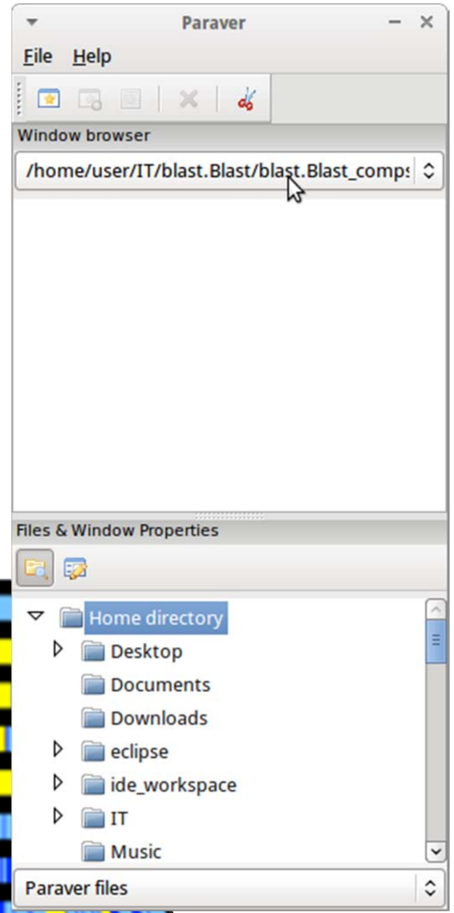
Runtime Monitoring

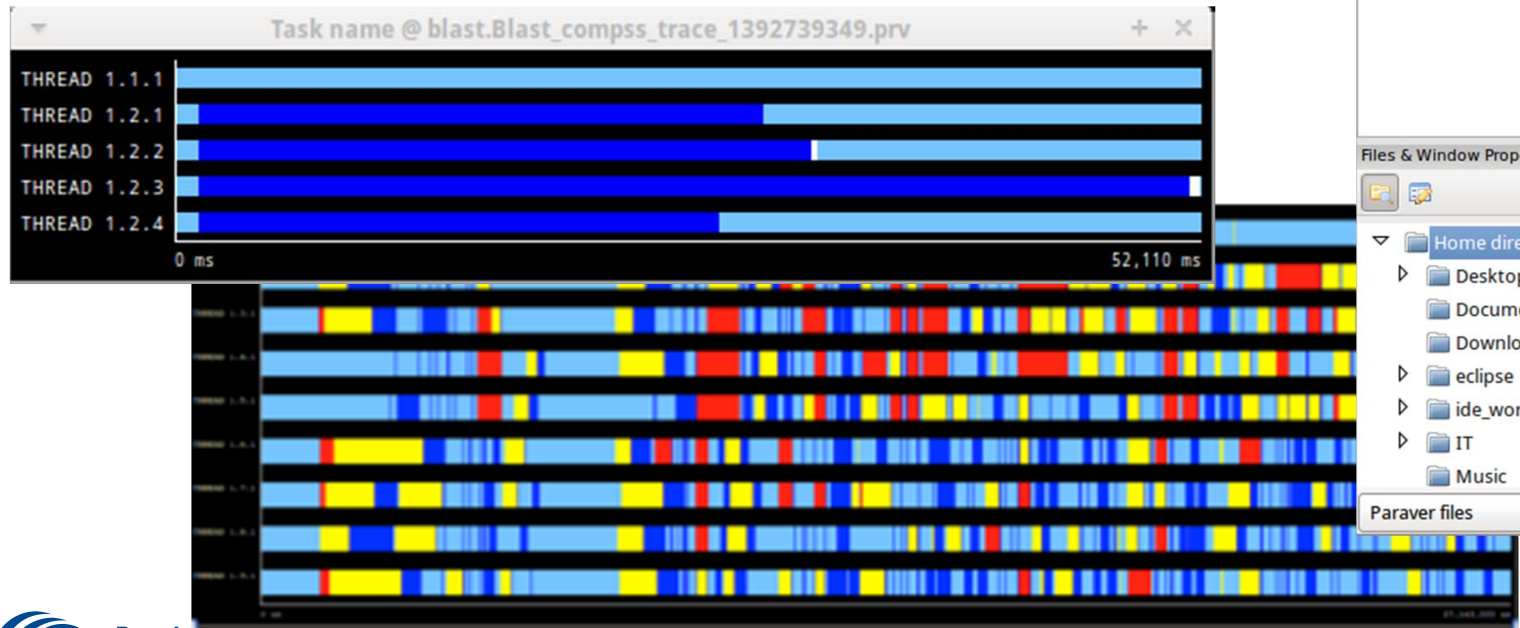
- The runtime of COMPSs provides some information at execution time so the user can follow the progress of the application:
  - Real-time monitoring information (http://localhost:8080/compss-monitor/)
    - # tasks
    - Resources usage information
    - Execution time per task
    - Real-time execution graph
    - …



Runtime features: Tracing and performance analysis

- Paraver is the BSC tool for trace visualization
  - Trace events are encoding in Paraver (.prv) format by Extrae
  - Paraver is a powerful tool for performance analysis
  - Paraver enables different views of a trace





Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Severo Ochoa project

- 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)

Severo Ochoa: Cloud and BigData

- Intersection between Cloud Computing and large scale data analytics/management
- Vertical approach integrating previous technologies
  - Programming environments and runtime systems
  - Resource management in heterogeneous systems and workloads
  - Storage architecture and management
- To be demonstrated with “in-house” scientific challenges

EXCELENCIA SEVERO OCHOA

Human Brain Project

What is the HBP?

- 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

Use Case 2: Developing new drugs for brain disorders

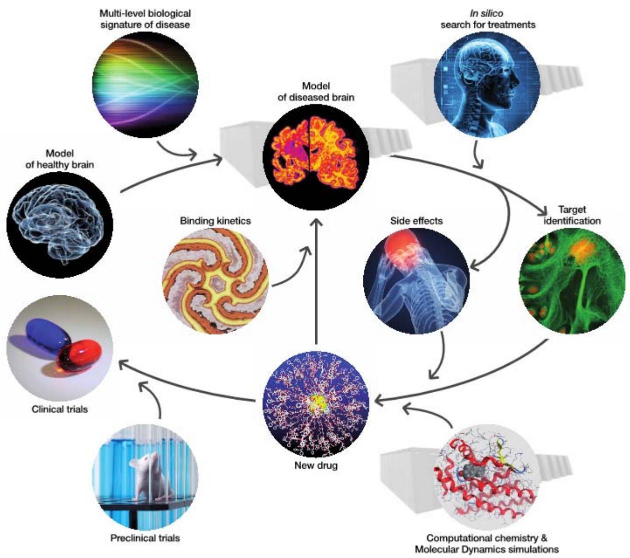
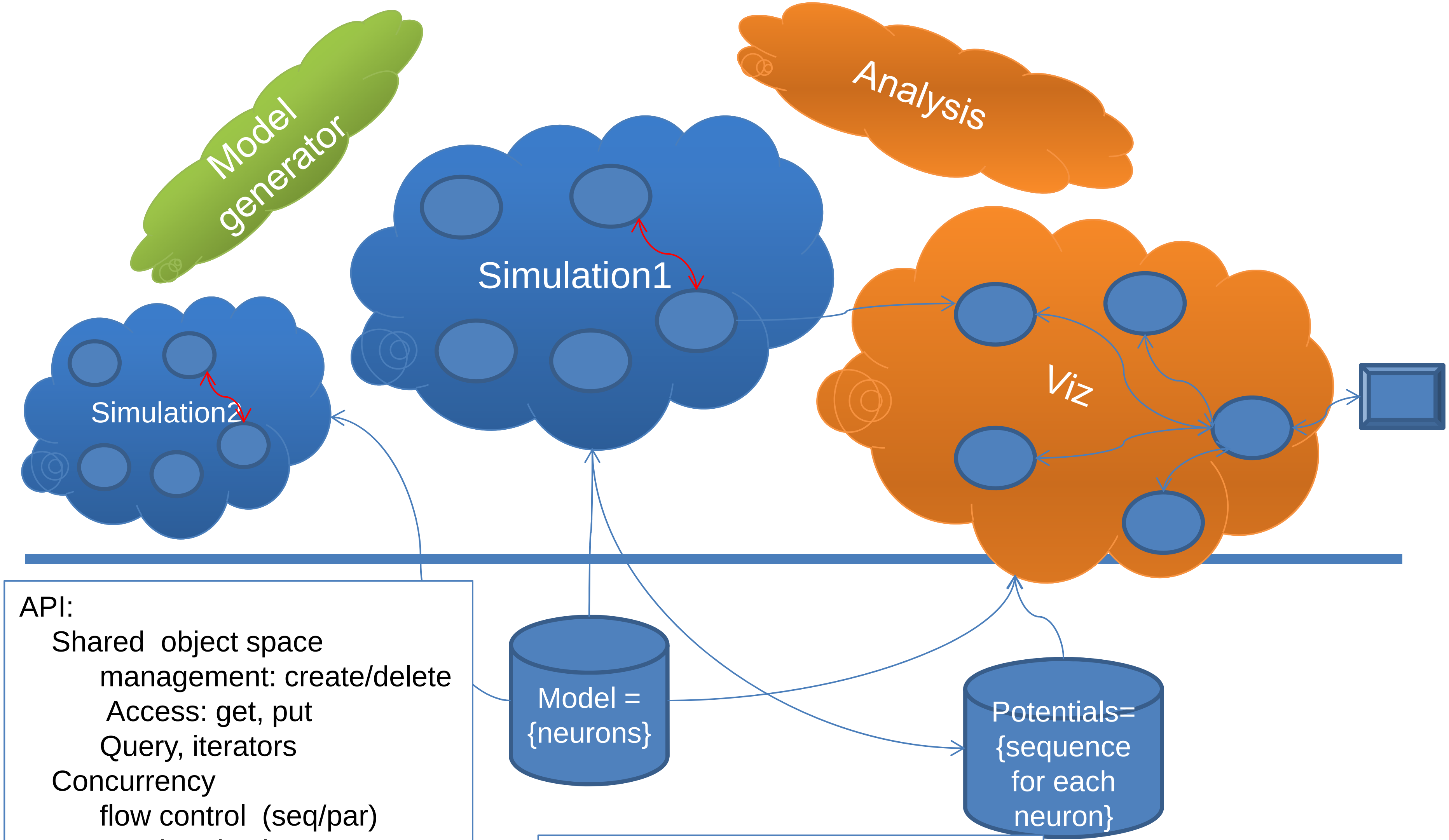


Figure 29: Use of the HBP platforms to accelerate drug development. Optimizing drug discovery and clinical trials

Human Brain Project

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

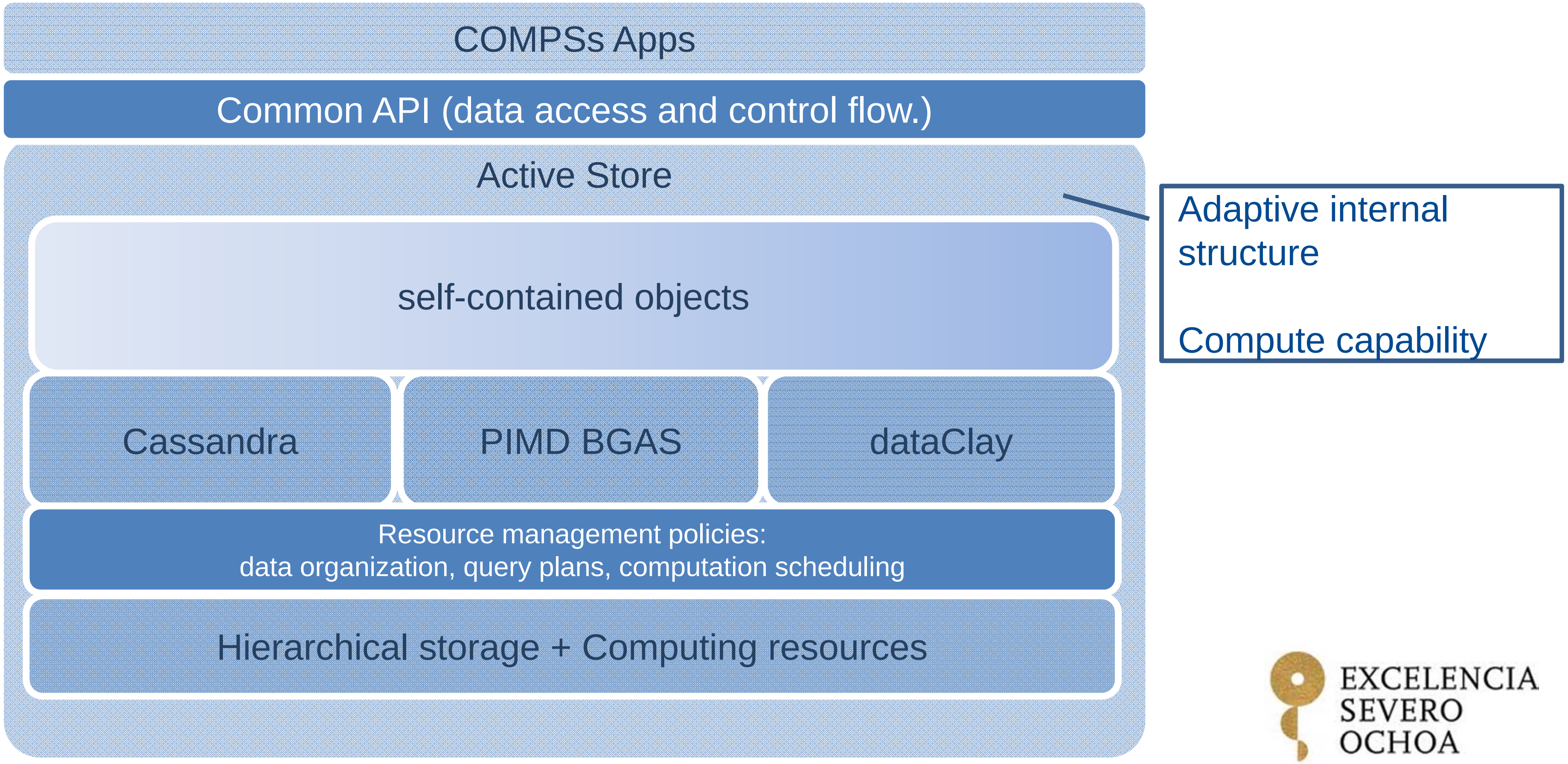
Sample scenarios



API:
- Shared object space
  - management: create/delete
  - Access: get, put
  - Query, iterators
- Concurrency
  - flow control (seq/par)
  - synchronization
- Consistency

Implementation:
- Persistent, Distributed, Resilient

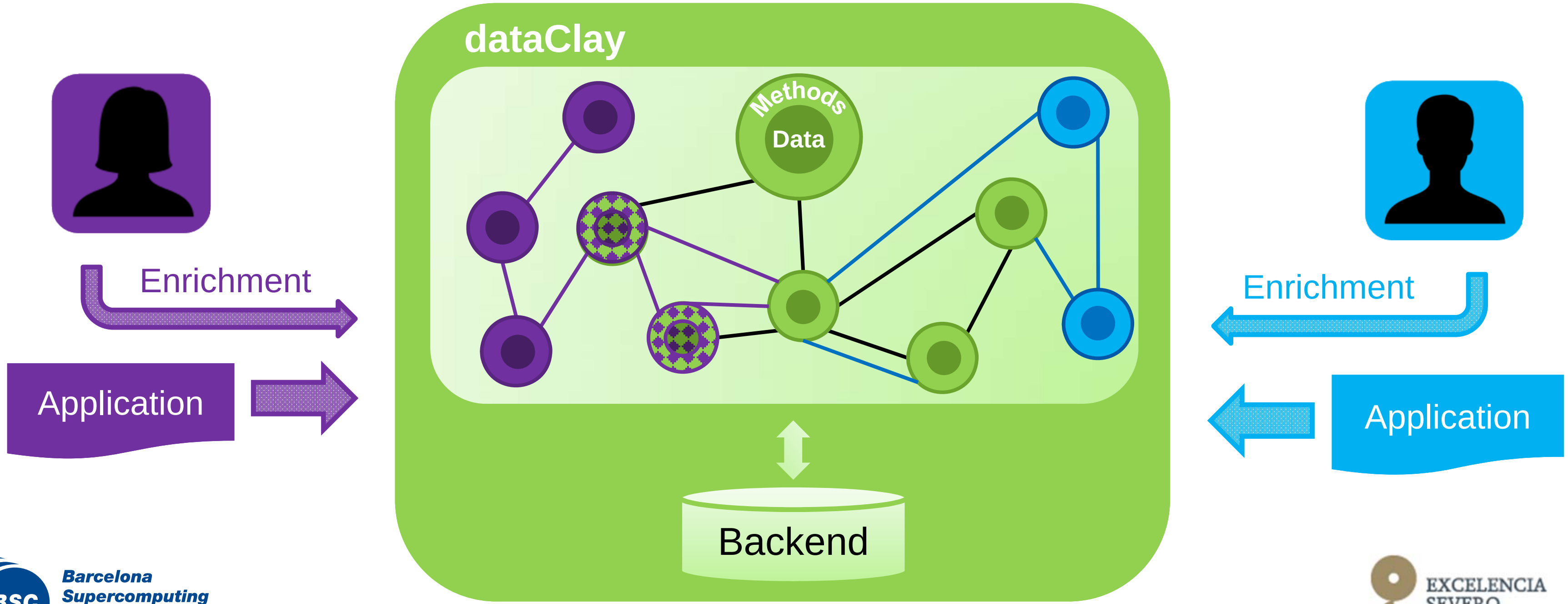
Architectural design of Active Storage



EXCELENCIA SEVERO OCHOA

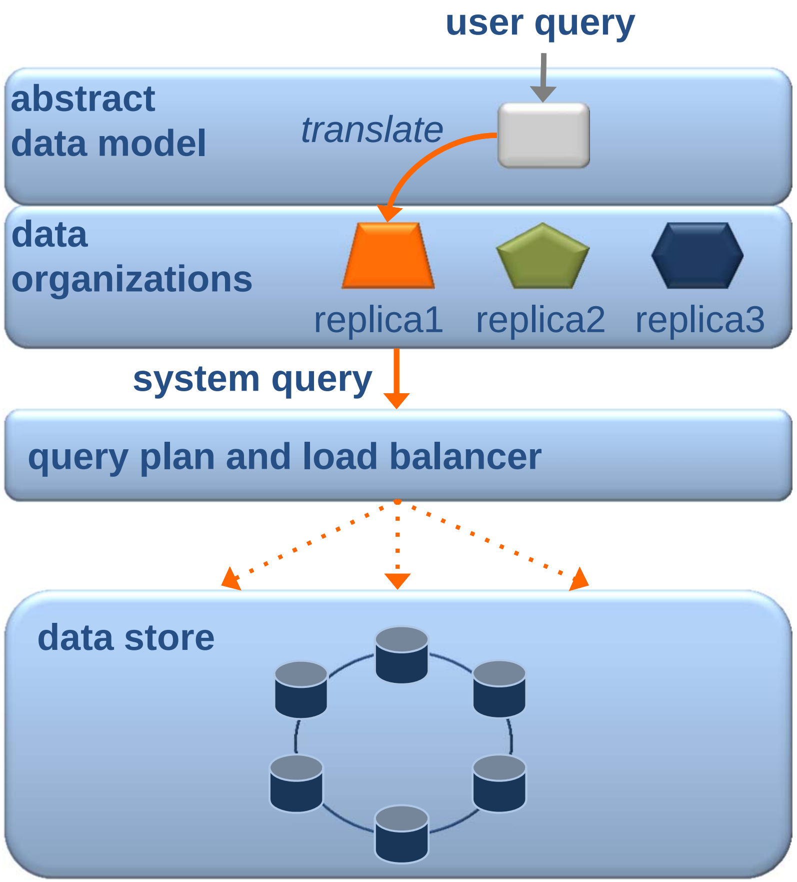
dataClay

- 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

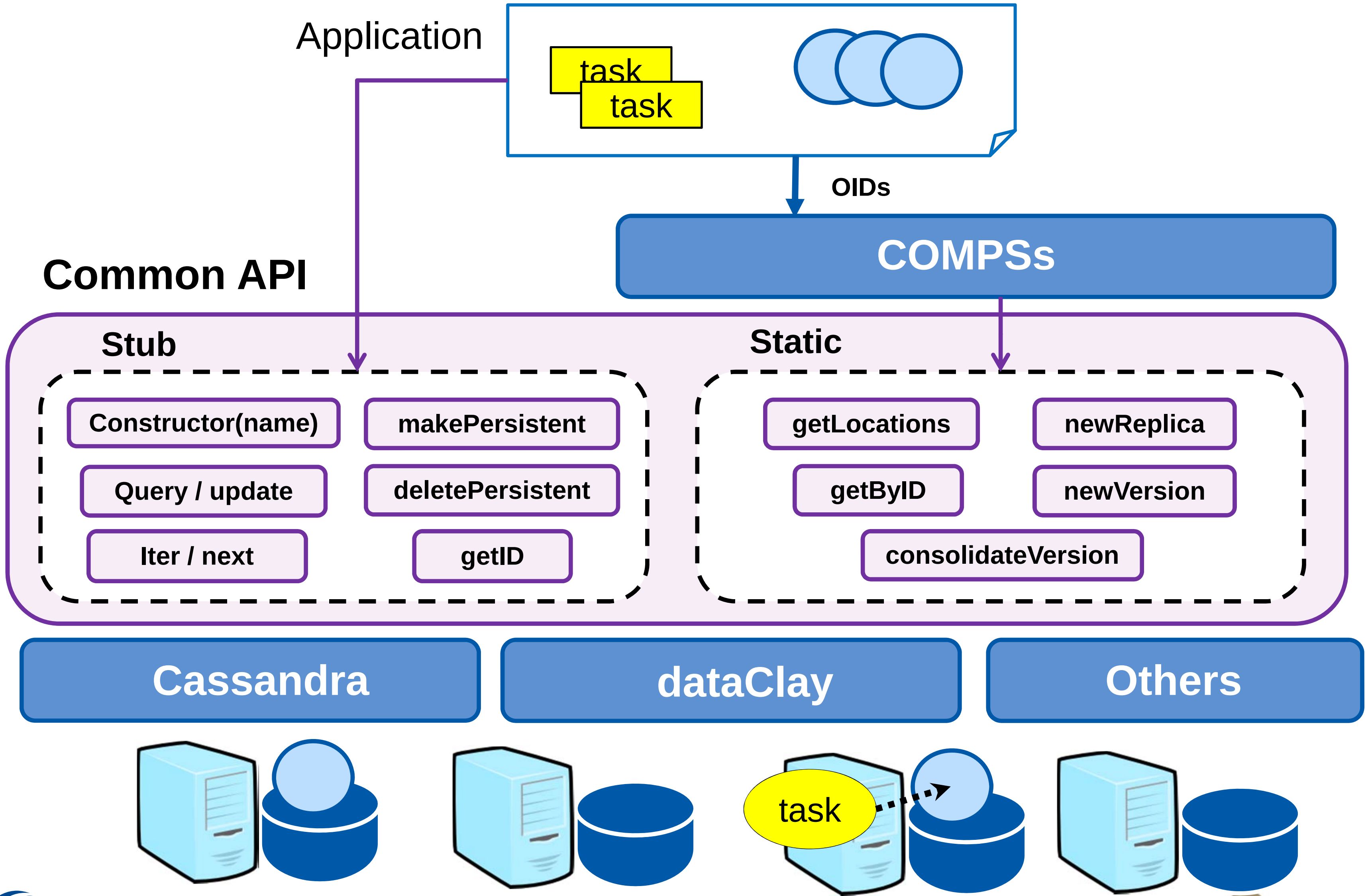


Bigdata resource management: overview

- Objectives:
  - to propose a highly-scalable resource management architecture for BigData applications
  - to decouple data modeling from data organization
  - to provide programmers with mechanisms to generate automatic data organization and automatic query code, that considers the performance of the data store system
- Apache Cassandra used to evaluate our proposals



Integration COMPSs – Common Storage API



Sample case: Neuroscience Data Processing

Find out correlation between spike trains with regard external events

Main program

```
import sys

neurons_file_name = sys.argv[1]
correlation_file_name = sys.argv[2]

nd = NeuronData(neurons_file_name)
correlation = Correlation()

 seed = 2398645
 delta = 1782324

for i in nd.spikes.keys():
  for j in nd.spikes.keys():
    if i<j:
      cc_surrogate_range(i, j, nd, …)
      seed = seed + delta

 dumpToFile(correlation, corr_file_name)
```

- load neuron data
- new object
- iterate over spikes keys
- store correlation data

Sequential code
Data in Python tables

Function definition

```
def cc_surrogate_range(i, j, nd, correlation, seed, num_surrs,
num_bins,
maxlag):
   sp = nd.spikes
   …

   for ni in range(i, i+1) :
     for nj in range(j, j+1):
       correlation.cc_originals[(ni,nj)] = correlate(sp[ni], sp[nj], …)
          …
       my_cc_surrs[:,0] = surrs_ij_sorted[round(num_surrs*0.95),:]
       my_cc_surrs[:,1] = surrs_ij_sorted[round(num_surrs*0.05),:]
       correlation.cc_surrs[(ni,nj)] = my_cc_surrs
```

- get access to spikes data
- read access
- write access

Neuroscience Data Processing @ PyCOMPSs + persistent objects

Main program

```
import sys

neuron_data_name = sys.argv[1]
correlation_name = sys.argv[2]

nd = NeuronData(neuron_data_name )
correlation = Correlation()
correlation.make_persistent(correlation_name)

 seed = 2398645
 delta = 1782324

for block_i in nd.spikes.keys():
   for block_j in nd.spikes.keys():
        cc_surrogate_range(block_i, block_j, nd, …)
         seed = seed + delta
```

get access to neuron data

new object

create persistent object

iterate over spikes keys

Tasks definition

```
from pycompss.api.task import task

@task()
def cc_surrogate_range(block_i, block_j, nd, correlation, seed,
num_surrs, num_bins, maxlag):

   sp = nd.spikes
   …

   for ni in block_i:
      for nj in block_j:
         if ni<nj:
             correlation.cc_originals[(ni,nj)] = correlate(sp[ni], sp[nj],
)
             …
         my_cc_surrs[:,0] = surrs_ij_sorted[round(num_surrs*0.95),:]
         my_cc_surrs[:,1] = surrs_ij_sorted[round(num_surrs*0.05),:]
         correlation.cc_surrs[(ni,nj)] = my_cc_surrs
```

get access to spikes data

load data

iterate over keys in blocks

store data

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Human Brain Project

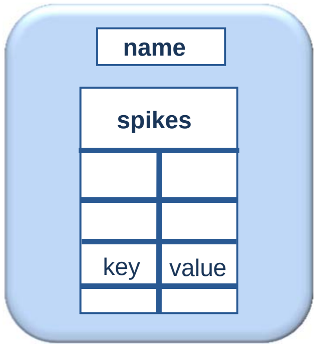
Neuroscience Data Processing @ PyCOMPSs and Cassandra: Data mapping

Programmer view

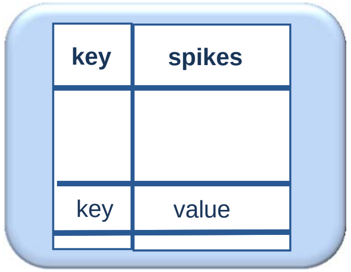
Backend (Cassandra)

```
class NeuronData
   name = string
   spikes = dict
```

NeuronData Class



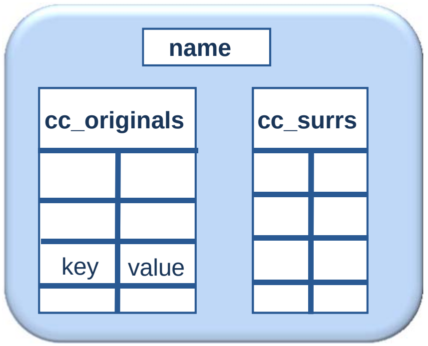
"Name1" Table



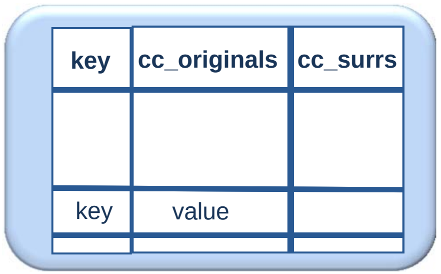
Data identifier in persistent storage

```
class Correlation
   name = string
   cc_originals = dict
   cc_surrs = dict
```

Correlation Class

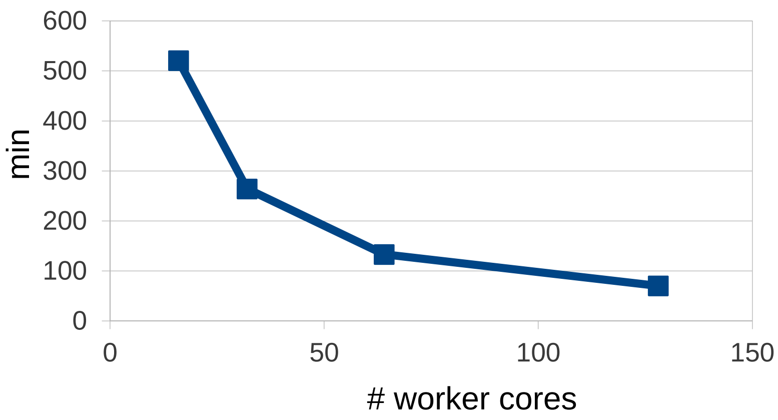


"Name2" Table

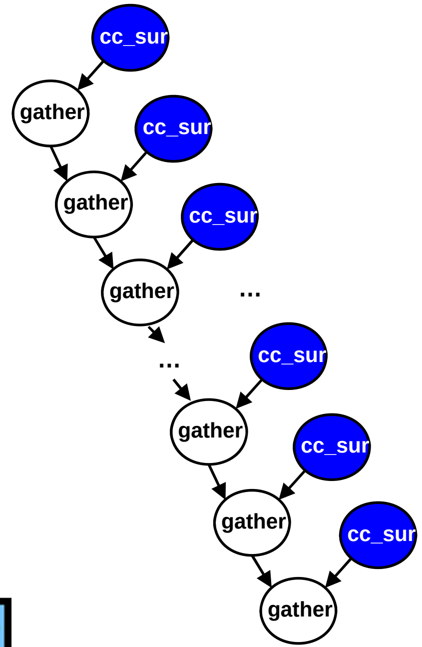


Neuroscience Data Processing: initial results

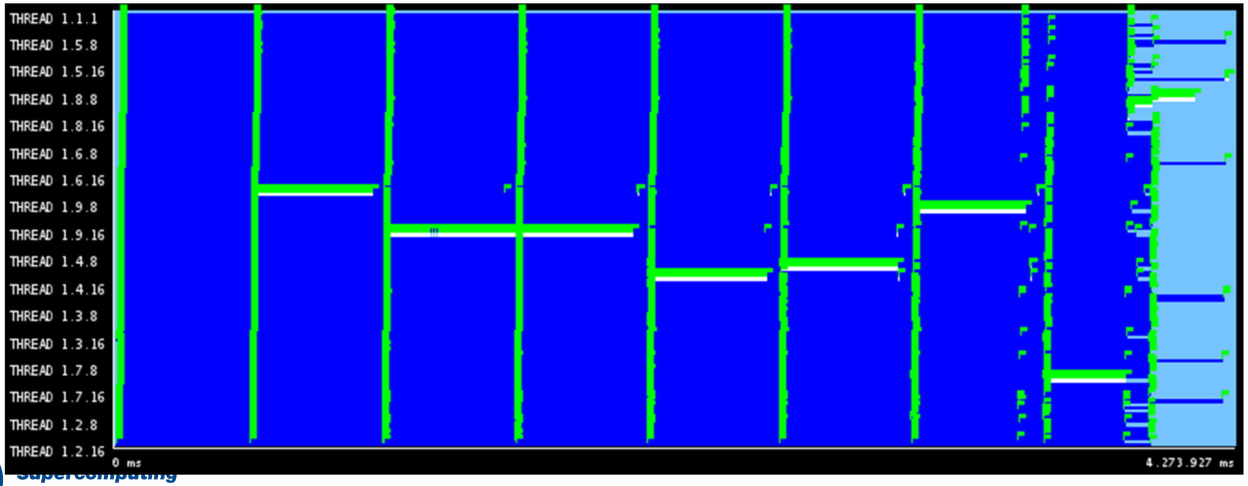
- Execution time



- Task dependency graph



- Execution trace



Projects where COMPSs is used/further developed

EURO SERVER

Human Brain Project

ASCETIC

EU Brazil Cloud Connect
EU Brazil Cloud Computing for Science

transPLANT

EXCELENCIA SEVERO OCHOA

Previous projects

Virtual multidisciplinary
EnviroNments USing
Cloud Infrastructures

Conclusions

- Sequential programming approach
- Parallelization at task level
- Transparent data management and remote execution
- Can operate on different infrastructures: Cluster, Grid, Cloud (Public/Private)
- Enables orchestration of Web services
- Demonstrated in several projects and applications
- New language bindings (Python) and extensions to integrate with new storage methodologies make it a promising environment for Big-data projects

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
- Pedro Benedicte (part time)
- Carlos Diaz
- Jorge Ejarque
- Fredy Juarez
- Daniele Lezzi
- Francesc Lordan
- Roger Rafanell
- Cristian Ramon (part time)
- Raul Sirvent
- Enric Tejedor

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

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**