Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

# Programming Distributed Computing Platforms with COMPSs

Pol Alvarez, Javier Alvarez, Ramon Amela, Rosa M. Badia, Javier Conejero, Marc Dominguez, Jorge Ejarque, Daniele Lezzi, Francesc Lordan, Cristian Ramon-Cortes, Sergio Rodriguez

Workflows & Distributed Computing Group

29-30/01/2019

Barcelona

# COMPSs Advanced Features


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Outline

- COMPSs and OmpSs Integration
- Integrating Binaries and MPI applications
- Execution Environments

# COMPSs - OmpSs Integration


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Programming Applications

- Implement application as a combination of two-level of tasks-based parallelism
  - Tasks: Parts of the code with a certain computation which can potentially be executed independently
    - Coarse-grain: distributed platform level ( ms)
    - Fine-grain: processor/device level (us/ms)
  - Main algorithm is implemented as a workflow of Coarse-grain tasks
  - Each coarse-grain tasks can be implemented as a workflow of fine-grain tasks
- Task versioning for enabling adaptation
  - Coarse-grain sequential or parallelized with fine-grain tasks, etc.
  - Fine-grain tasks implemented for different target devices

# Programming Example

Main Code

```
int main( int argc, char **argv ){
  Matrix A, B, C ;
  …

  //Calculate Matrix Multiplication
  for( int i=0; i<N; i++) {
    for( int j=0; j<N; j++) {
      for( int k=0; k<N; k++) {
        multiply(A.block[i][k],B.block[k][j],
                 C.block[i][j]));
      }
    }
  }
}
```

Coarse-grain task Definition

```
interface Matmul
{
  @Constraints(processors={@Processor(ProcessorType=CPU, ComputingUnits=4)});
  void Block::multiply(in Block b1, in Block b2, inout Block b3);

  @Constraints(processors={@Processor(ProcessorType=GPU, ComputingUnits=1)});
  @Implements(Block::multiply);
  void Block::multiplyGPU( in Block b1, in Block b2, inout Block b3);
};
```

Coarse-grain task Implementations

```
void Block::multiply(Block b1, Block block2, Block block3) {
  for (int i=0; i<M; i++) {
    for (int j=0; j<M; j++) {
      #pragma omp task inout (b3.data[i][j]) in(b1.data[i*M;M], b2.data[0;M*M])
      for (int k=0; k<M; k++) {
        b3.data[i*M + j] += b1.data[i*M + k] * b2.data[k*M + j];
      }
    }
  }
  #pragma omp taskwait
}
```

Fine-grain task Definition

Fine-grain task Implementation

```
void Block::multiplyGPU(Block b1, Block block2, Block block3) {
  int NB = M/SB_SIZE; //GPU sub-block size
  for (int i=0; i<NB; i++) {
    for (int j=0; j<NB; j++) {
      for (int k=0; k<NB; k++) {
        Muld(b1.data[i*NB+k], b2.data[k*NB+j], b3.data[i*NB+j], NB);
      }
    }
  #pragma omp taskwait
}
```

```
//Kernel.h
#pragma omp target device(cuda) ndrange(2, 64, 64, 32, 32)
#pragma omp task in(A[0:NB*NB], B[0:NB*NB]) inout(C[0:NB*NB])
__global__ void Muld(double* A, double* B, double* C, int NB);
```

# Other Languages

- Python

```
@constraints(Processors[type=CPU, computing
@ompss(binary="ompss_app")
@task(file_in=FILE_IN, file_out=FILE_OUT)
def my_OmpSs_task(file_in, file_out):
    pass

__main__:
    my_OmpSs_task("file_in.txt", "file_out.txt")
```

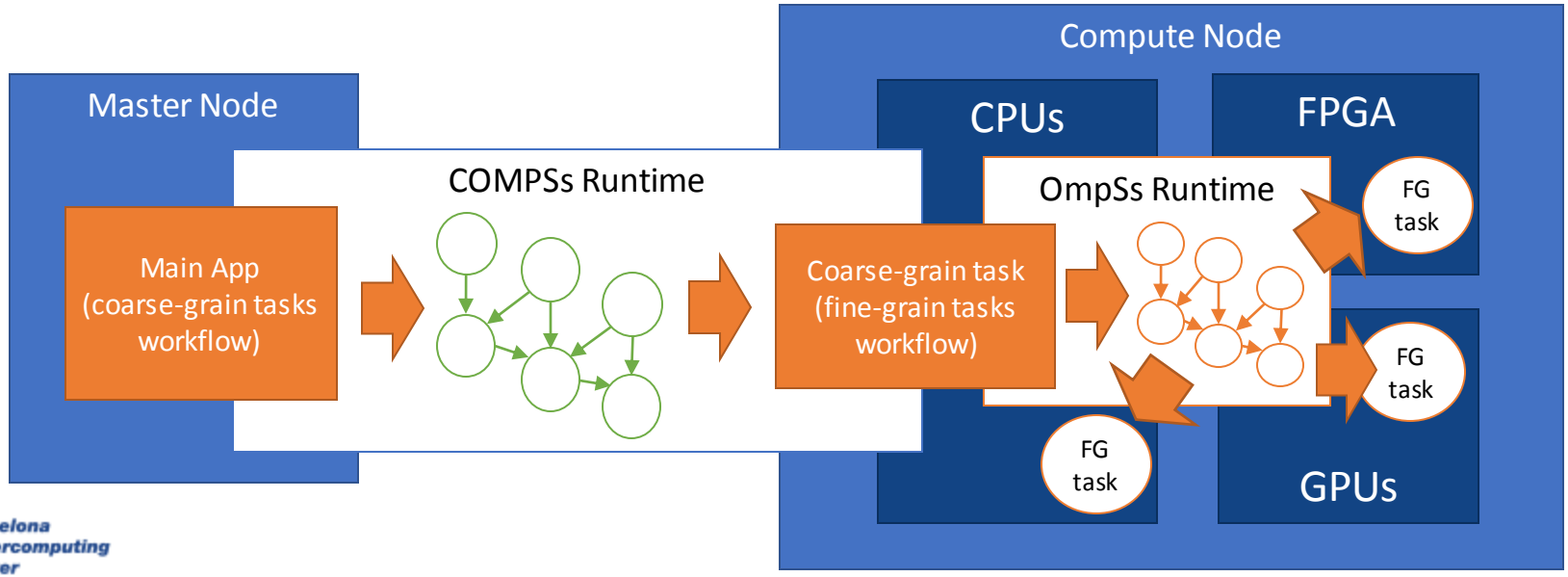
- Java

```
public interface OmpSsExampleItf {
    @OmpSs(binary = "ompss_app")
    int ompssTask(
        @Parameter() String message,
        @Parameter(type = Type.FILE, direction = Direction.OUT) String fOut
    );
}
```

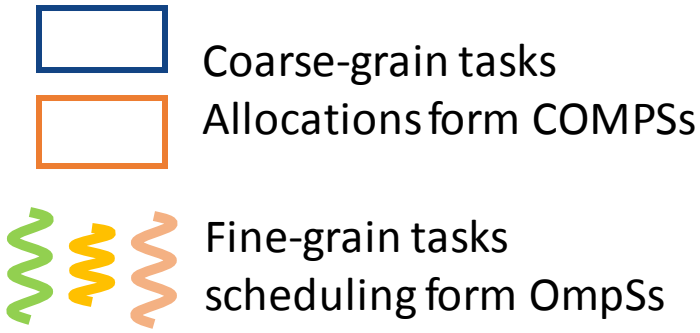
```
public static void main(String[] args) {
     String msg = "Hello World!";
     String outputFile = "output.txt";
     int ev = OMPSS.ompssTask(msg, outputFile);
}
```

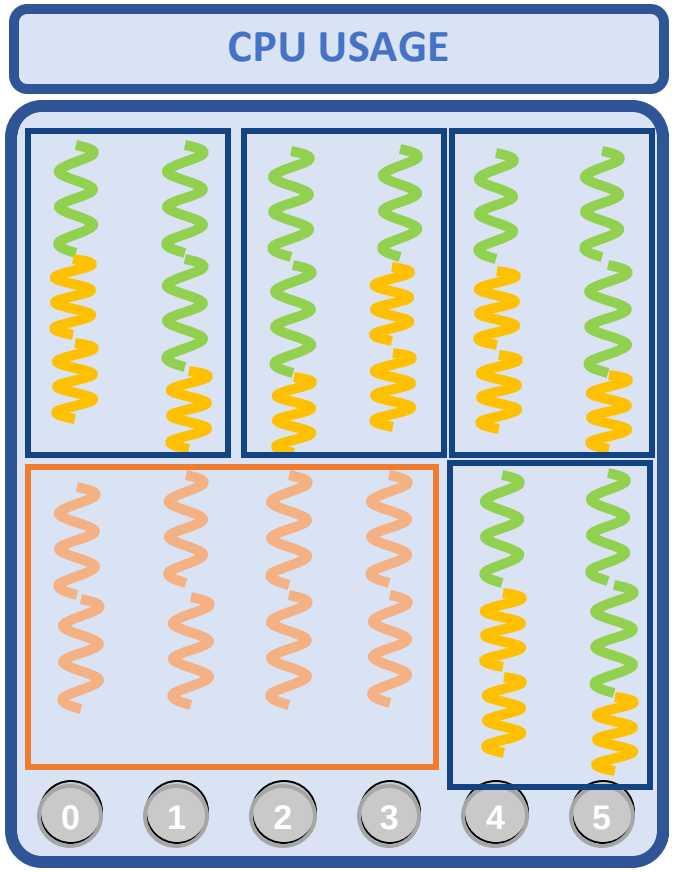
# Runtime Environment

- Parallelism and execution managed at two levels:
  - Platform level (COMPSs)
    - Assigns an execute coarse-grain tasks in computing nodes.
    - Data locality and transfers between computing nodes
    - Configures the node runtime
  - Intra-node level (OmpSs)
    - manages the intra-node heterogeneity
    - Assigns and execute fine-grain tasks in node computing devices
    - Data locatity and transfers between main memory and device memory



# Intra-node CPU management



# Compilation & Execution

- Compilation
  - Generate master/worker stubs (C++ do not support reflection)
  - Command:
    - compss_build_app --ompss [--cuda|--opencl<appName>
- Execution
  - Same as normal C COMPSs application
    - runcompss master/<appName> [app_args]

# Integrating Binaries and MPI applications with COMPSs


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")
  - @mpi(binary="mpiApp.bin", runner="mpirun", computingNodes=8)
  - @ompss(binary="ompssApp.bin")
- Can be combined with the @constraint and @implement decorators

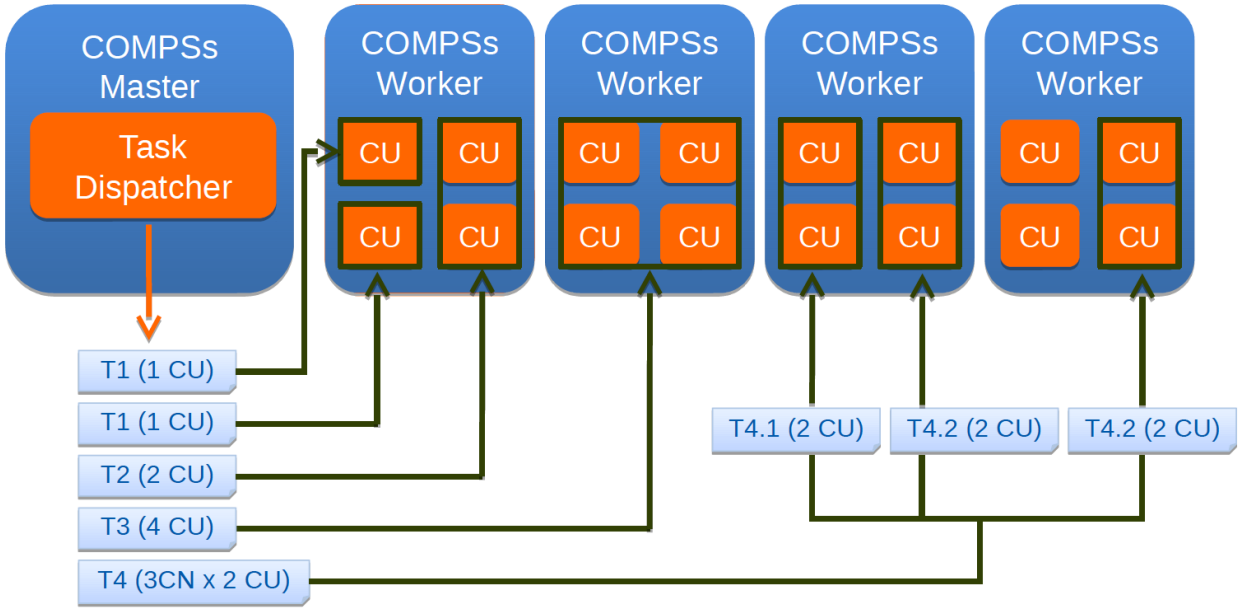
```
@binary(binary="app.bin", workingDir="/myApp")
@task()
def func(l):
    pass
```

# Support for MPI tasks

- Extension of the PyCOMPSs interface
- Resource manager aware of multi-node tasks

```
@constraint (computingUnits= "2")
@mpi (runner="mpirun",
      computingNodes= "3”,
      ...)
@task (returns=int, stdOutFile=FILE_OUT_STDOUT, stdErrFile=FILE_OUT_STDERR)
def nems(stdOutFile, stdErrFile):
    pass
```

Launches execution in
3 processes
2 threads / node



# Support for MPI tasks

In [ ]:
```
from pycompss.api.mpi import mpi
```

In [ ]:
```
computing_units = "2"
computing_nodes = 1
```

In [ ]:
```
@constraint(ComputingUnits=computing_units)
@mpi(runner="mpirun", binary="${GMX_BIN}/gmx_mpi", computingNodes=computing_nodes)
@task(input_tpr_path=FILE_IN, output_gro_path=FILE_OUT, output_trr_path=FILE_OUT, output_edr_pa
      output_log_path=FILE_OUT)
def mdrun_pc(mdrun="mdrun",
             s="-s", input_tpr_path="",
             c="-c", output_gro_path="",
             o="-o", output_trr_path="",
             x="-x", output_xtc_path="",
             e="-e", output_edr_path="",
             g="-g", output_log_path="",
             nt="-nt", nt_value=0):
    pass

# The task call is:
# mdrun_pc(input_tpr_path="", output_gro_path="", ...)

# The task execution will automatically call:
# mpirun -np 2 -hostfile X gmx_mpi mdrun -s itp -c ogp ...
```

# Support for binaries

In [ ]:
```
from pycompss.api.binary import binary
```

In [ ]:
```
@binary(binary="${GMX_BIN}/gmx")
@task(input_gro_path=FILE_IN, output_gro_path=FILE_OUT)
def editconf_pc(editconf="editconf",
                f="-f", input_gro_path="",
                o="-o", output_gro_path="",
                d="-d", distance_to_molecule="",
                bt="-bt", box_type="cubic",
                c="-c"):
    pass

# The task call is:
# editconf_pc(input_gro_path="", output_gro_path="", distance_to_molecule="")

# The task execution will automatically call:
# gmx editconf -f igp -o ogp -d dtm -bt cubic -c
```

In [ ]:
```
from pycompss.api.parameter import Type, Prefix

@binary(binary="gnuplot")
@task(plotscript_path=FILE_IN, output_png_path={Prefix: "#"})
def gnuplot_pc_image(plotscript_path="", output_png_path=""):
    pass

# The task call is:
# gnuplot_pc_image(plotscript_path="", output_png_path="")

# The task execution will automatically call:
# gnuplot plotscript_path

# The prefix can also be used for parameters of the form: --x=value
```

# Hands-on

- Example of @binary
  - 7_Binary.ipynb

# Integrating Binaries (Java)

```
public interface SampleItf {
  @Binary(binary = "/path/to/binary")
  void binaryTask(
    @Parameter(type = Type.STRING, direction = Direction.IN) String message,
    @Parameter(type = Type.FILE, direction = Direction.IN, prefix="–in=") String fileIn,
    @Parameter(type = Type.FILE, direction = Direction.OUT, prefix="–out") String fileOut,
    @Parameter(type = Type.FILE, direction = Direction.OUT, stream = Stream.STDERR) String fileErr
 );
 // command: /path/to/binary message –in=fileIn –out=fileOut 2>fErr
}
```

```
import binary.BINARY;
…
  public static void main(String[] args) {
    //Binary Task invocation
    BINARY.binaryTask("message", "fileIn", "fileOut", "fileErr");
    …
  }
```

```
package binary;
public class BINARY {
  public static void binaryTask( String message,
    String fileIn, String fileOut, String fileErr){
      /* Dummy implementation, just to compile*/
    }
}
```

# Integrating MPI (Java)

```
public interface SampleItf {
  @MPI(binary = "/path/to/binary", mpiRunner = "mpirun", computingNodes = "2")
  @Constraints(computingUnits = "2")
  void mpiTask(
    @Parameter(type = Type.STRING, direction = Direction.IN) String opt1,
    @Parameter(type = Type.FILE, direction = Direction.OUT, prefix="–out") String fileOut
  );
 // command: mpirun -np 4 -H node1,node1,node2,node2 /path/to/binary opt1 -out=fileOut
}
```

```
import binary.BINARY;
…
  public static void main(String[] args) {
    // MPI Task invocation
    MPI.mpiTask("option1", "fileOut");
    …
  }
```

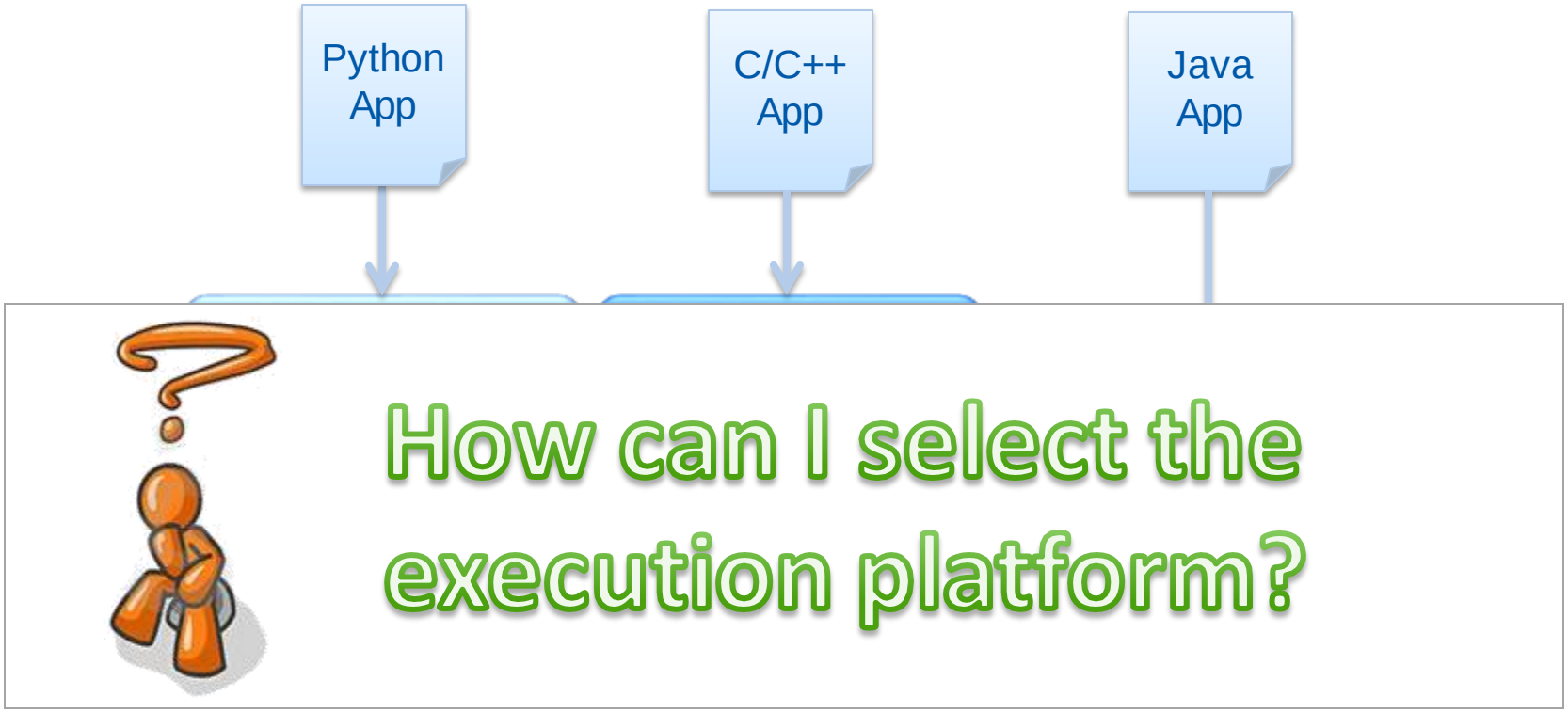
```
package mpi;
public class MPI{
  public static void mpiTask(String opt1, String fOut){
   /* Dummy Implementation just to compile */
  }
}
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

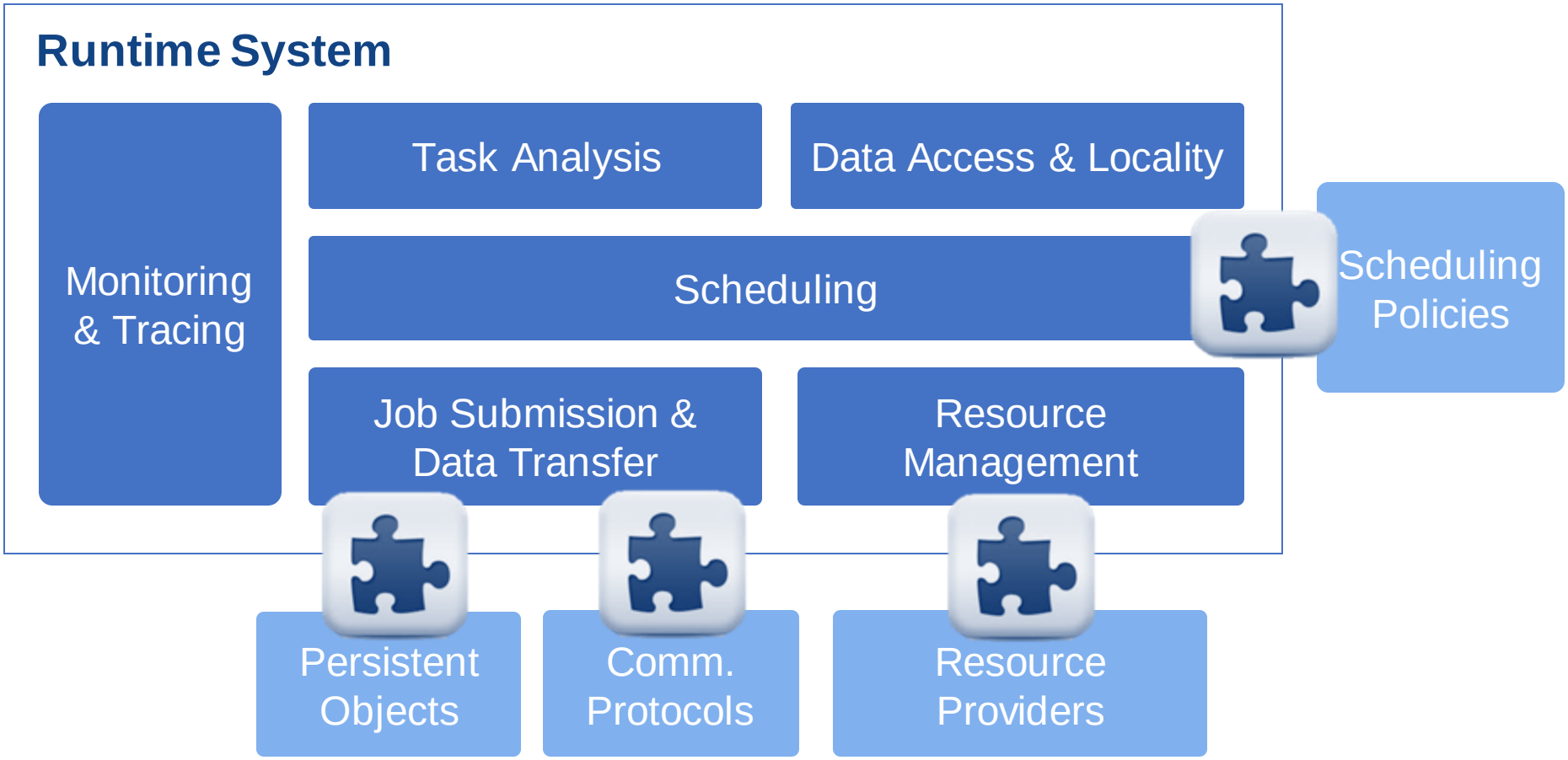
# COMPSs Execution Environments


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# COMPSs Architecture



# Runtime Extensions



# Execution Environments Configuration

**Specifies:**
- Resources and Project files
- Scheduler, Comm. Adaptor
- Persistent object storage

runcompss options

**Infrastructure Description**
- Describe the available resource in the infrastructure
- Describe Cloud Providers: Images and VM Templates

## Runtime System

Exec. Mngmt & Data Transfers

Persistent Object Storage: DataClay, Hecuba, Redis

Communication Adaptor: NIO, GAT

Schedulers

Resources

Cloud Connector: jClouds, rOCCI

Slurm, Docker, Mesos

resources.xml

project.xml

**Master-Worker Comm. Mechanism**
- GAT: Restrictred environments (only ssh access) and Grid Middleware
- NIO: Efficient Persistent workers implementation in controlled and secured environments

**Resource Scalability**
- Provide interaction with resource providers to create and destroy new computing resources

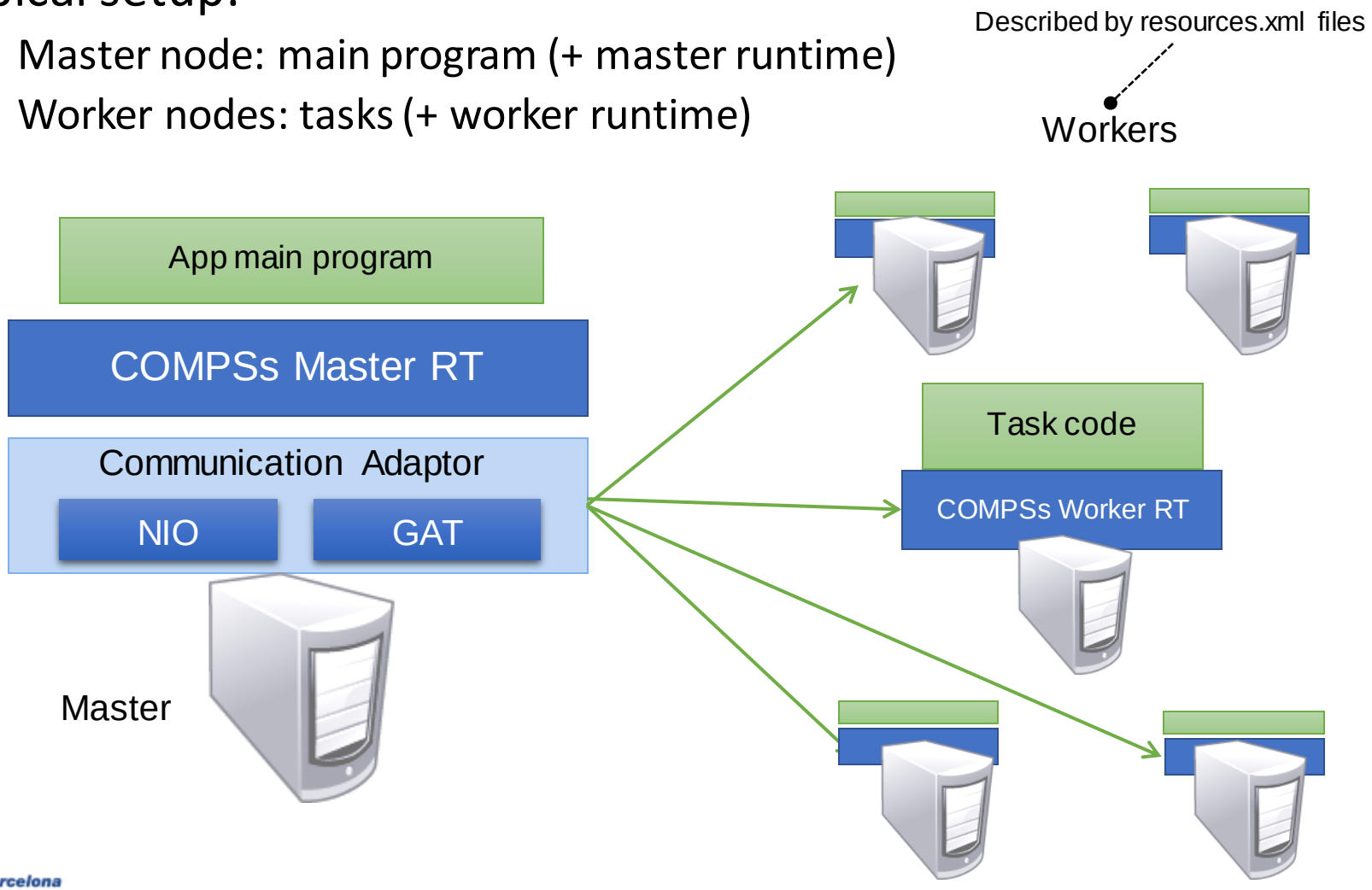
**Application Exec. Desc.**
- Selection of resources
- Application Code Location
- Working directory
- Provided as execution command argument

# Basic Execution Environments

- Interactive Computing Nodes
- Clusters (interaction with batch jobs systems)
- Clouds (interaction with Cloud Provider APIs)

# COMPSs @ Interactive Hosts

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



# Configuration: Resources Specification

- Resources.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ResourceList>
   <!--Description for any physical node-->
   <ComputeNode Name="172.20.200.18">
     <Processor Name="P1">
        <ComputingUnits>4</ComputingUnits>
        <Architecture>amd64</Architecture>
        <Speed>3.0</Speed>
     </Processor>
     <Memory>
        <Size>256.2</Size>
        <Type>Non-volatile</Type>
     </Memory>
     <Storage>
        <Size>2000.0</Size>
     </Storage>
      <OperatingSystem>
        <Type>Linux</Type>
        <Distribution>OpenSUSE</Distribution>
        <Version>13.2</Version>
     </OperatingSystem>
                  …
     <Software>
        <Application>Java</Application>
        <Application>Python</Application>
     </Software>
     <Adaptors>
          <Adaptor Name="integratedtoolkit.nio.master.NIOAdaptor">
             <SubmissionSystem>
                <Interactive/>
             </SubmissionSystem>
             <Ports>
                <MinPort>43001</MinPort>
                <MaxPort>43002</MaxPort>
             </Ports>
          </Adaptor>
     </Adaptors>
   </ComputeNode>

   <ComputeNode Name="172.20.200.19">
        …
   </ComputeNode>
</ResourceList>
```

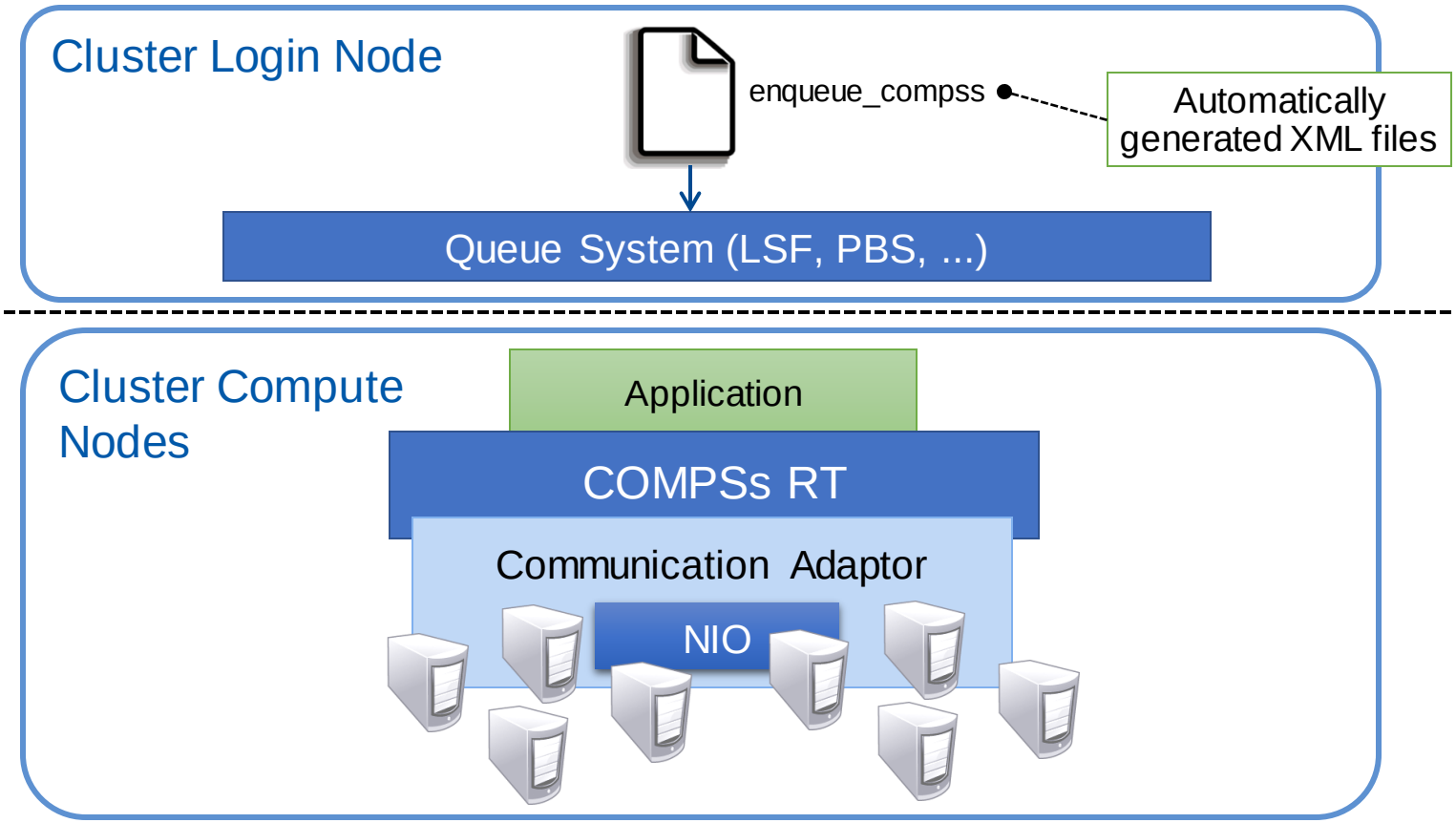
# Configuration: Project Specification

- Project.xml

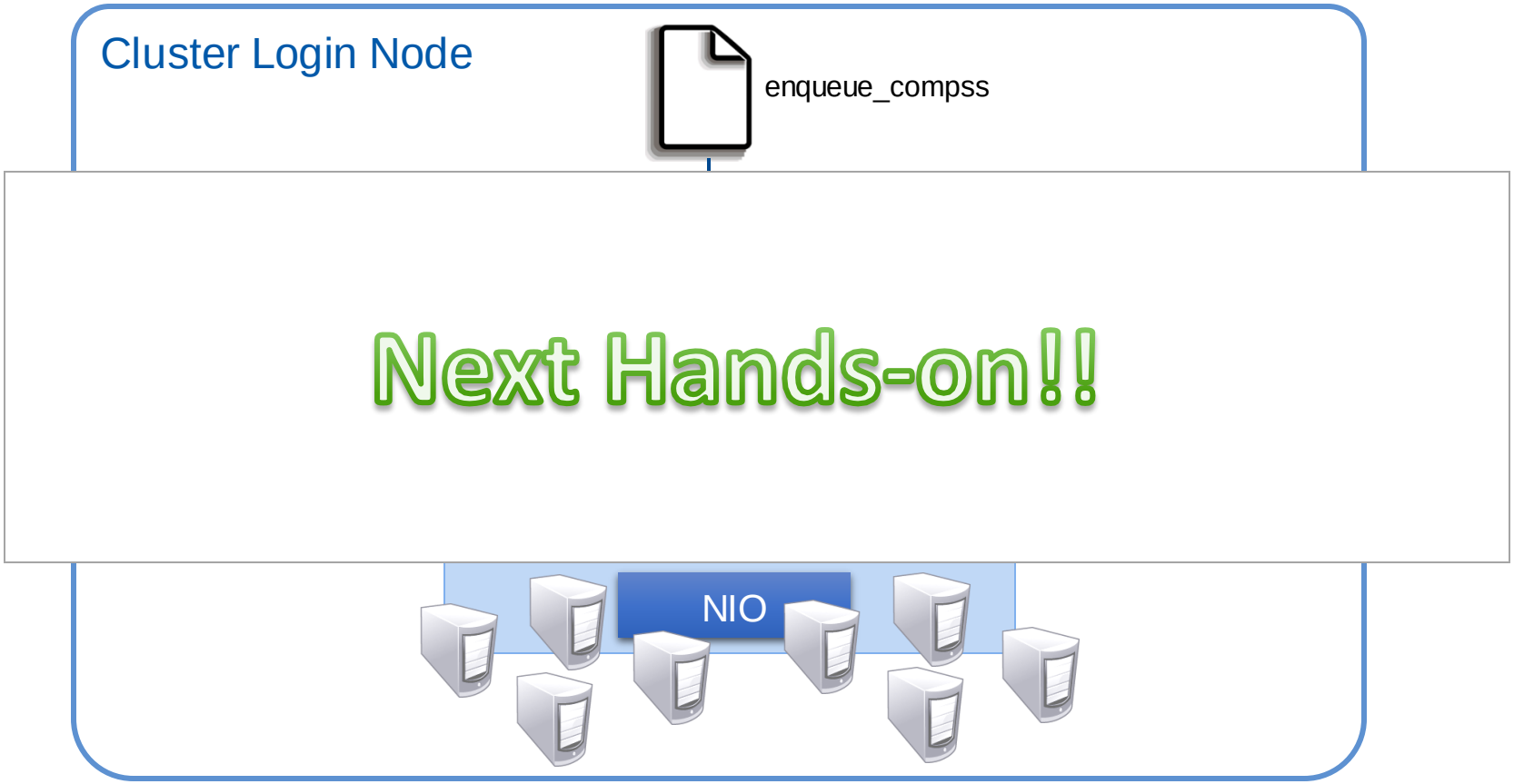
```
<?xml version="1.0" encoding="UTF-8"?>
<Project>
    <!--Description of used nodes in an application and where is the application installed-->
    <ComputeNode Name="172.20.200.18">
        <InstallDir>/opt/COMPSs/</InstallDir>
        <WorkingDir>/tmp/</WorkingDir>
        <Application>
            <AppDir>/home/user/apps/app_A/</AppDir>
            <LibraryPath>/home/user/apps/app_A/lib</LibraryPath>
            <Classpath>/home/user/apps/app_A/clases/</Classpath>
            <Pythonpath>/home/uthser/apps/app_A/clases/py<Pythonpath>
        </Application>
    </CompuNode>

    <ComputeNode Name="172.20.200.19">
      …
    </ComputeNode>
    ….
</Project>
```

# COMPSs@Cluster

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

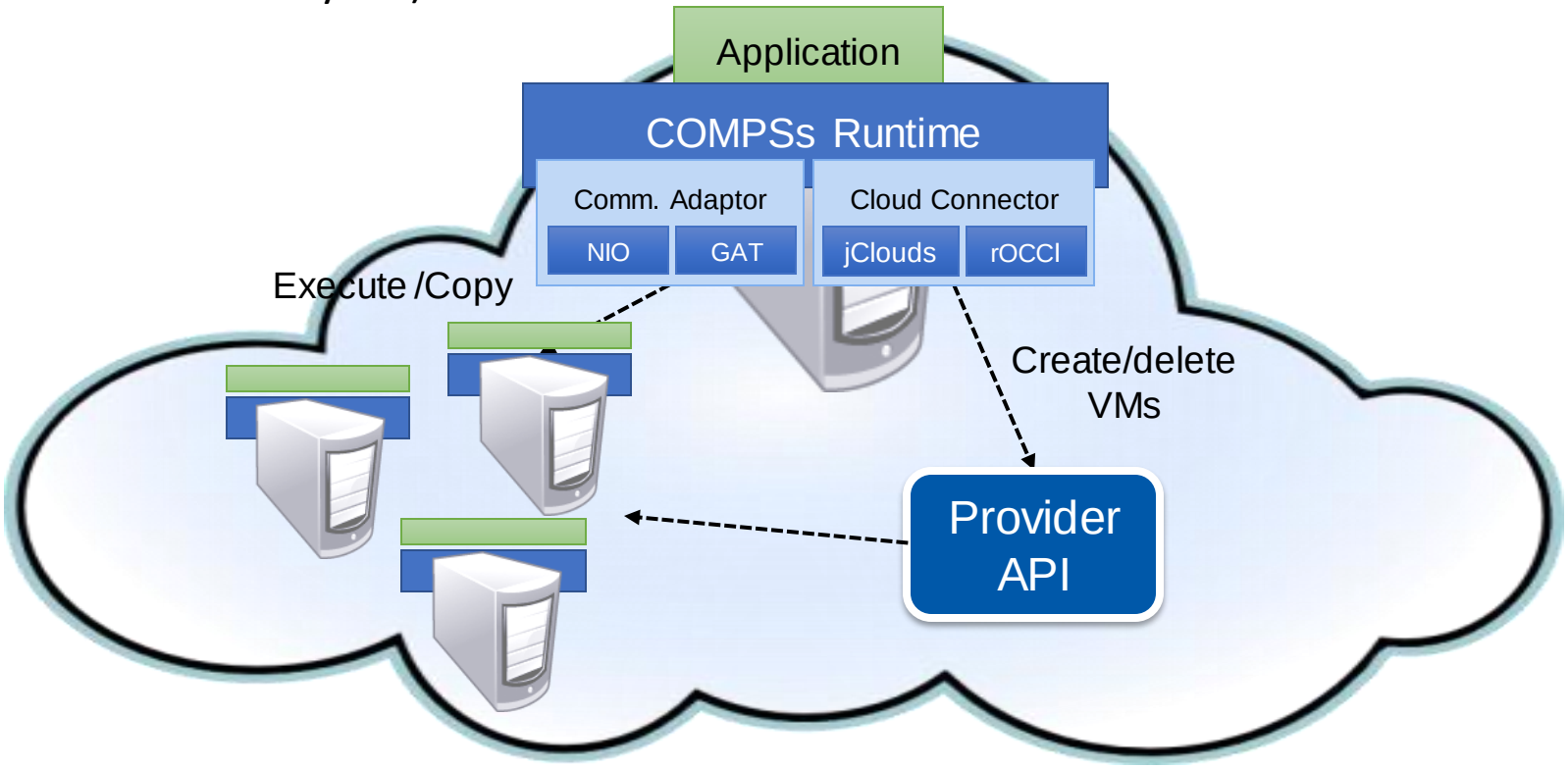


# COMPSs@MN



# COMPSs@Cloud

- Execution of COMPSs applications in Clouds
  - Select de connector to interact the Cloud provider
  - Adaptor to communicate VMs (NIO if provider supports firewall management, GAT if only ssh)



# COMPSs@Cloud

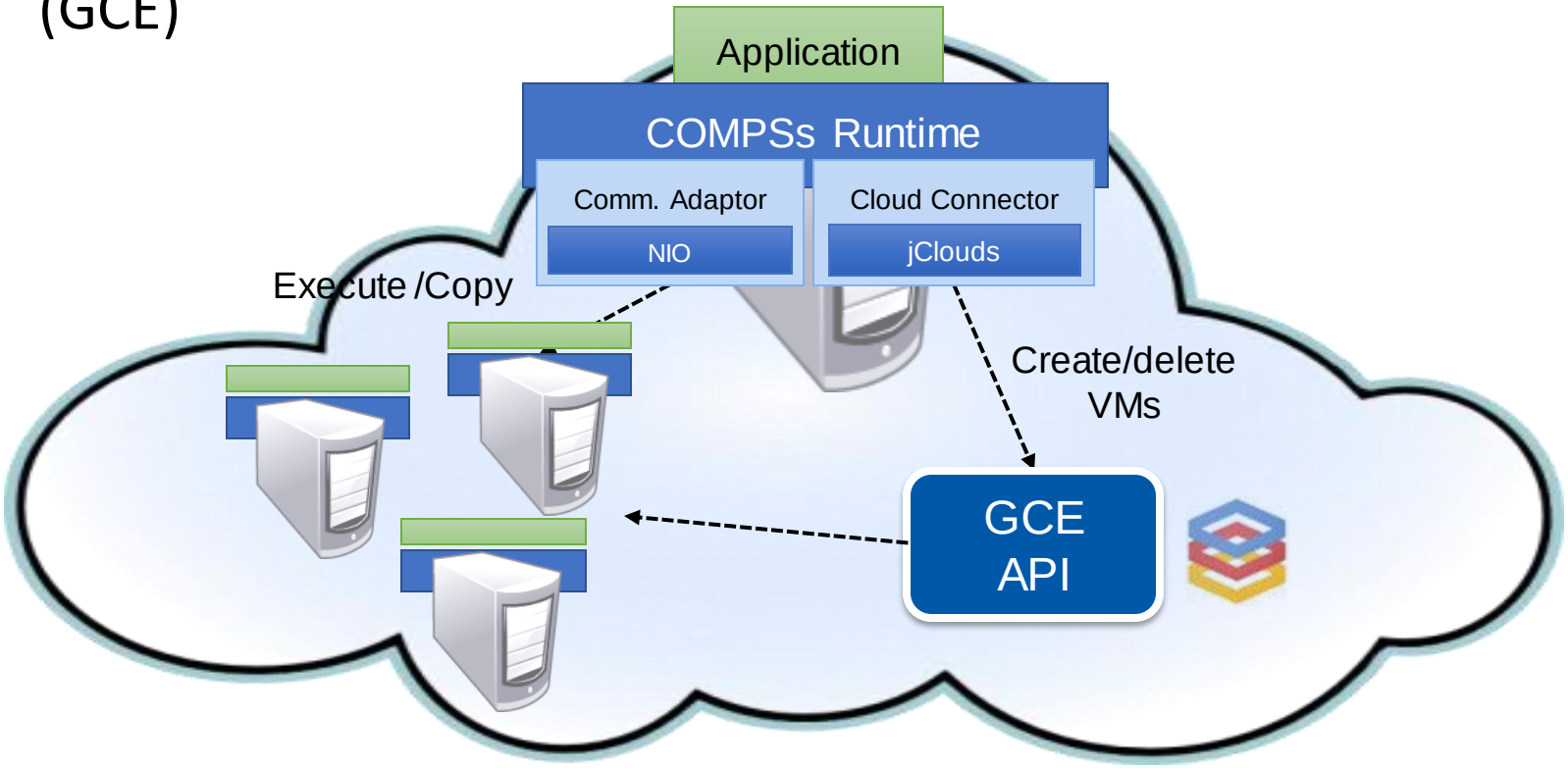
```
<ResourceList>
  <CloudProvider name="BSCCloud">
    <Endpoint>
      <Server>https://bscgrid20.bsc.es:11443</Server>
      <ConnectorJar>conn-rocci.jar</ConnectorJar>
      <ConnectorClass>es.bsc.conn.rocci.ROCCI</ConnectorClas
    </Endpoint>
    <Images>
      <Image name="debianbase">
        <CreationTime>120</CreationTime>
        <Adaptors>…
        <OperatingSystem>...
        <Software>…
      </Image>
        ..
    </Images>
    <InstanceTypes>
        <InstanceType Name="bsc.small">
      <Processor>...
      <Memory>…
       </InstanceType>
         …
    </InstanceTypes>
  </CloudProvider>
</ResourceList>
```

```
<Project>
    <Cloud>
      <InitialVMs>0</InitialVMs>
      <minVMCount>2</minVMCount>
      <maxVMCount>5</maxVMCount>
      <Provider name="BSCCloud">
        <LimitOfVMs>5</LimitOfVMs>
        <Property>
          <Name>user-cred</Name>
          <Value>/home/.../cert.pem</Value>
        </Property>
        <Property>
          <Name>user</Name>
          <Value>userbsc</Value>
        </Property>
        <ImageList>
          <Image name="debianbase">
            <InstallDir>/opt/COMPSs/</InstallDir>
            <WorkingDir>/tmp/</WorkingDir>
            <Package>
              <Source>/home/…/AppName.tar.gz</Source>
              <Target>/home/user/</Target>
             </Package>
          </Image>
        </ImageList>
        <InstanceTypes>
          <InstanceType name="bsc.small"/>
        </InstanceTypes>
      </Provider>
    </Cloud>
</Project>
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# DEMO: COMPSs@Google

- Execution of COMPSs applications in Google Compute Engine (GCE)



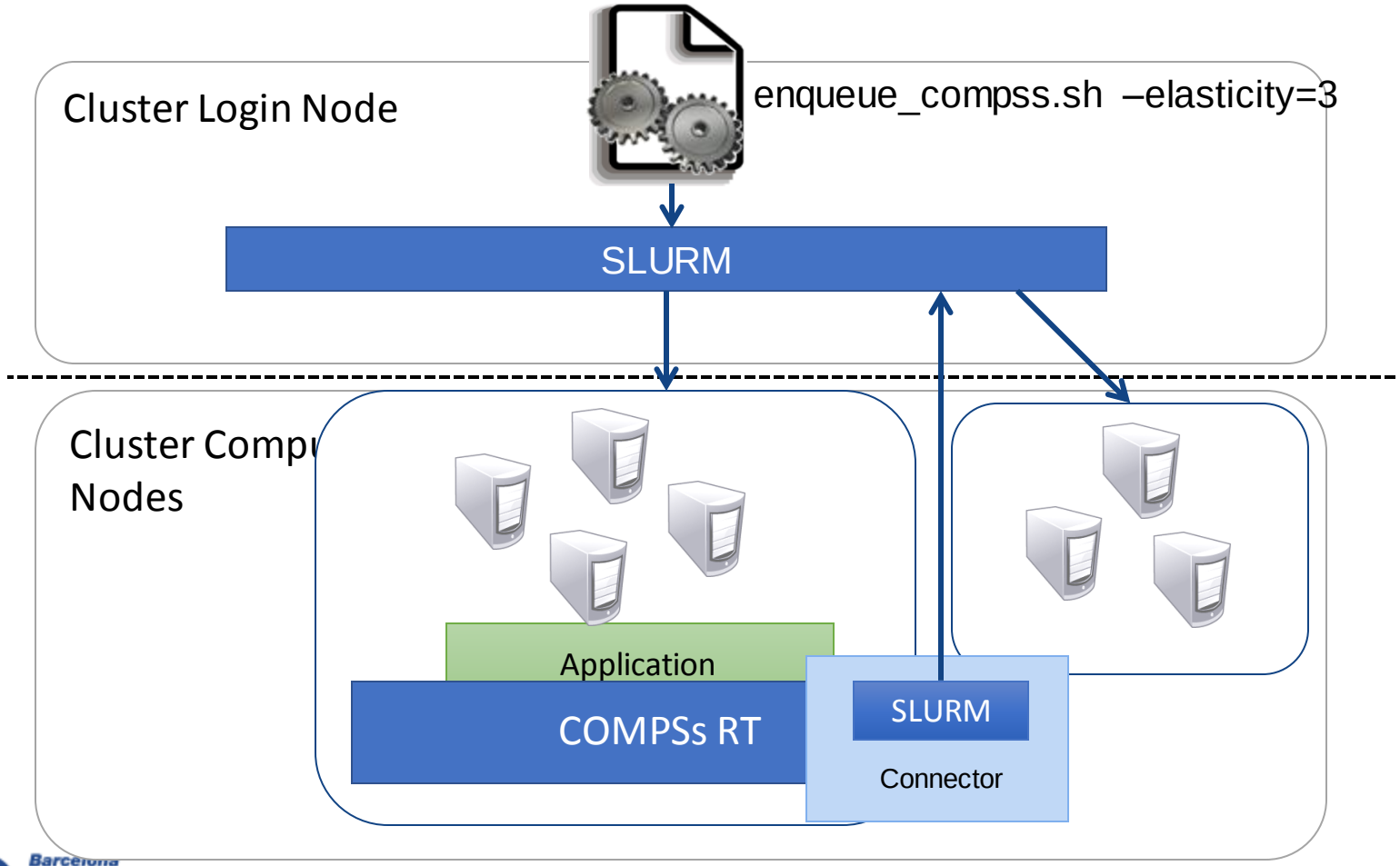

https://www.youtube.com/watch?v=XGaqUje_2zY

# Advanced Execution Environment

- Elasticity @ Clusters (SLURM Connector)
- Container engines
  - Docker
  - Singularity
- Cloud bursting
- Multi-grids

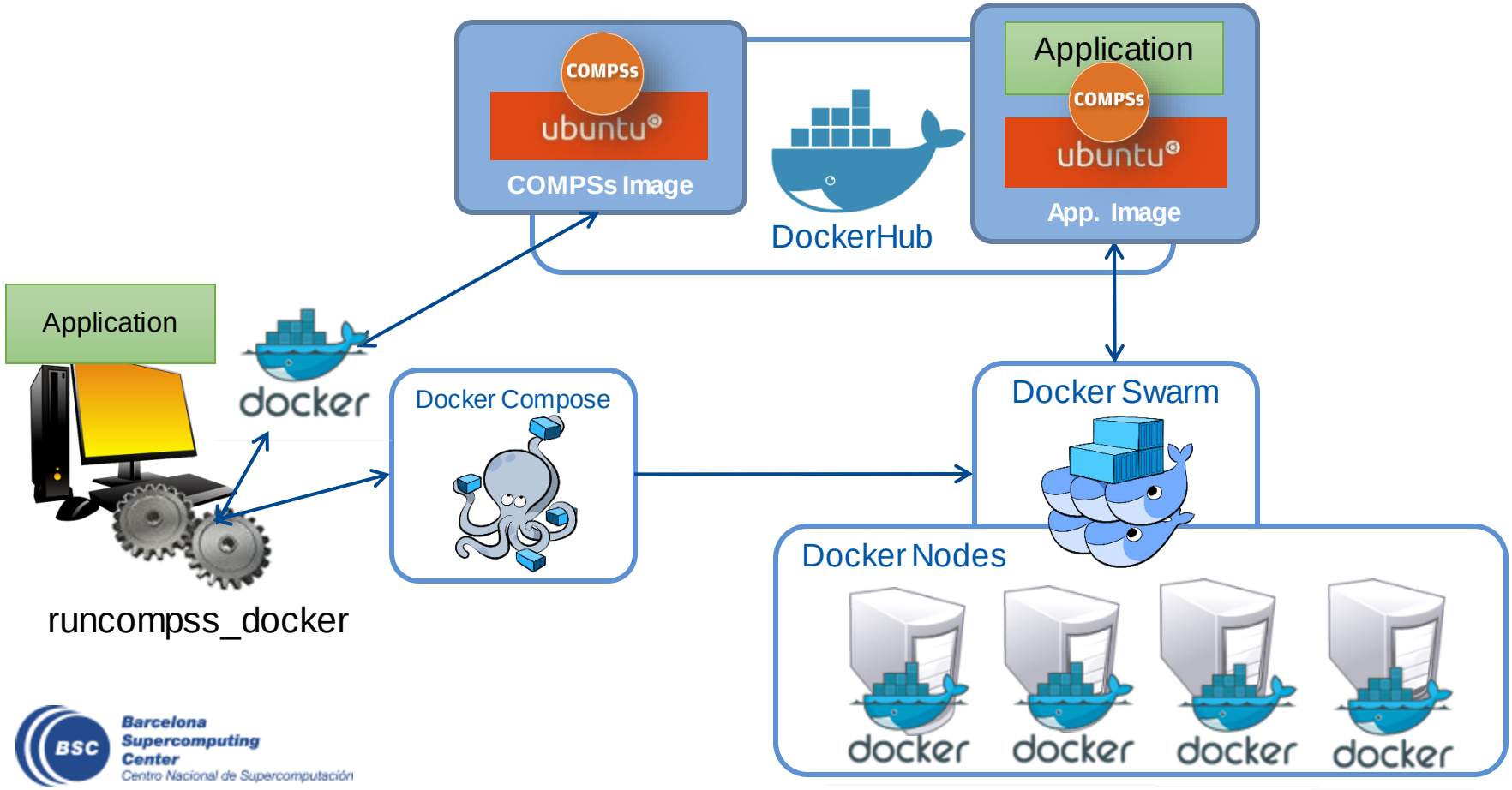
# Elasticity@Clusters with SLURM Connector

- Enable the SLURM connector at submission time



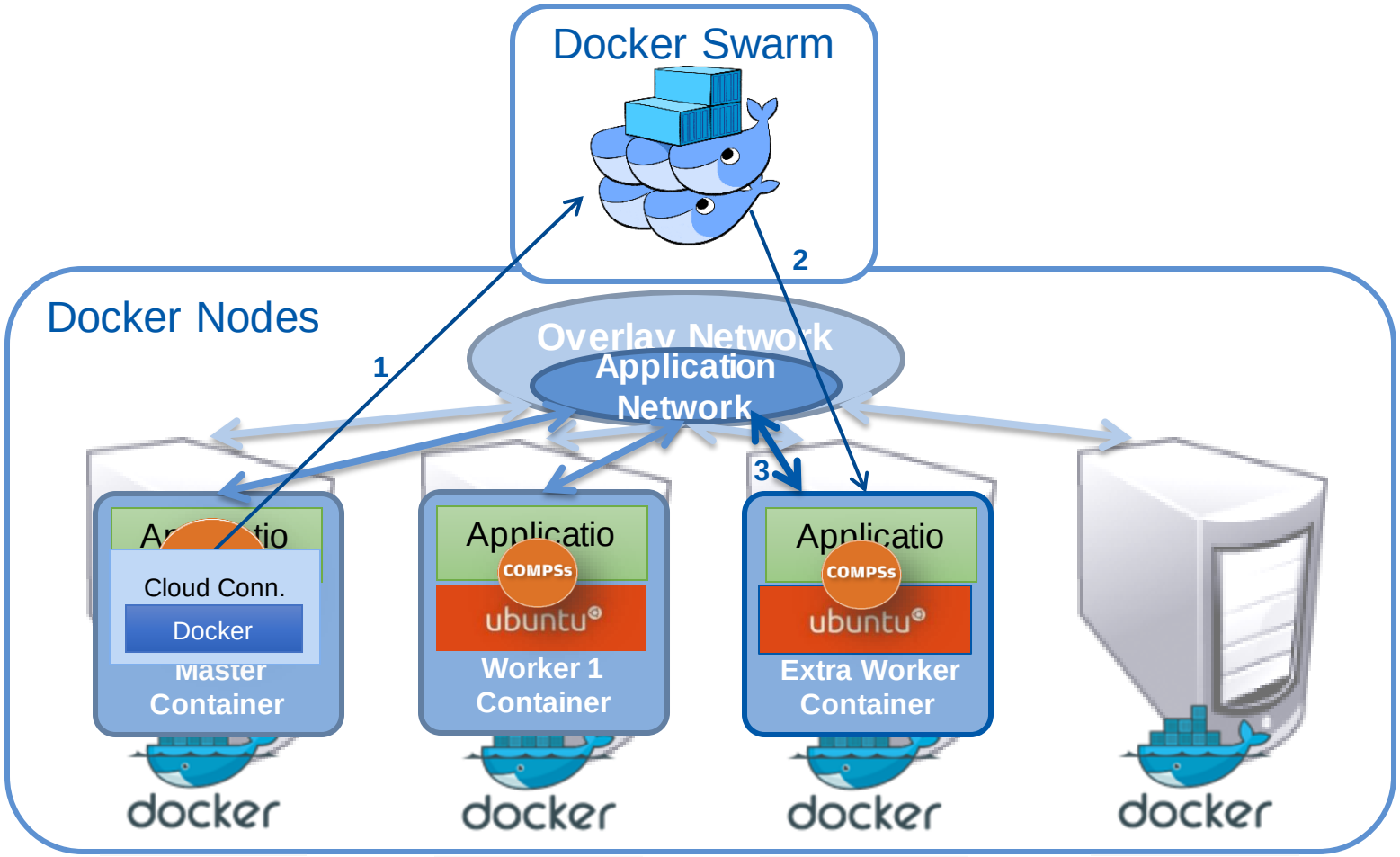
# COMPSs@Docker

- Keep as transparent for the user as posible
  - Same as running a local compss application (runcompss command)
- Deploy applications as a set of docker container



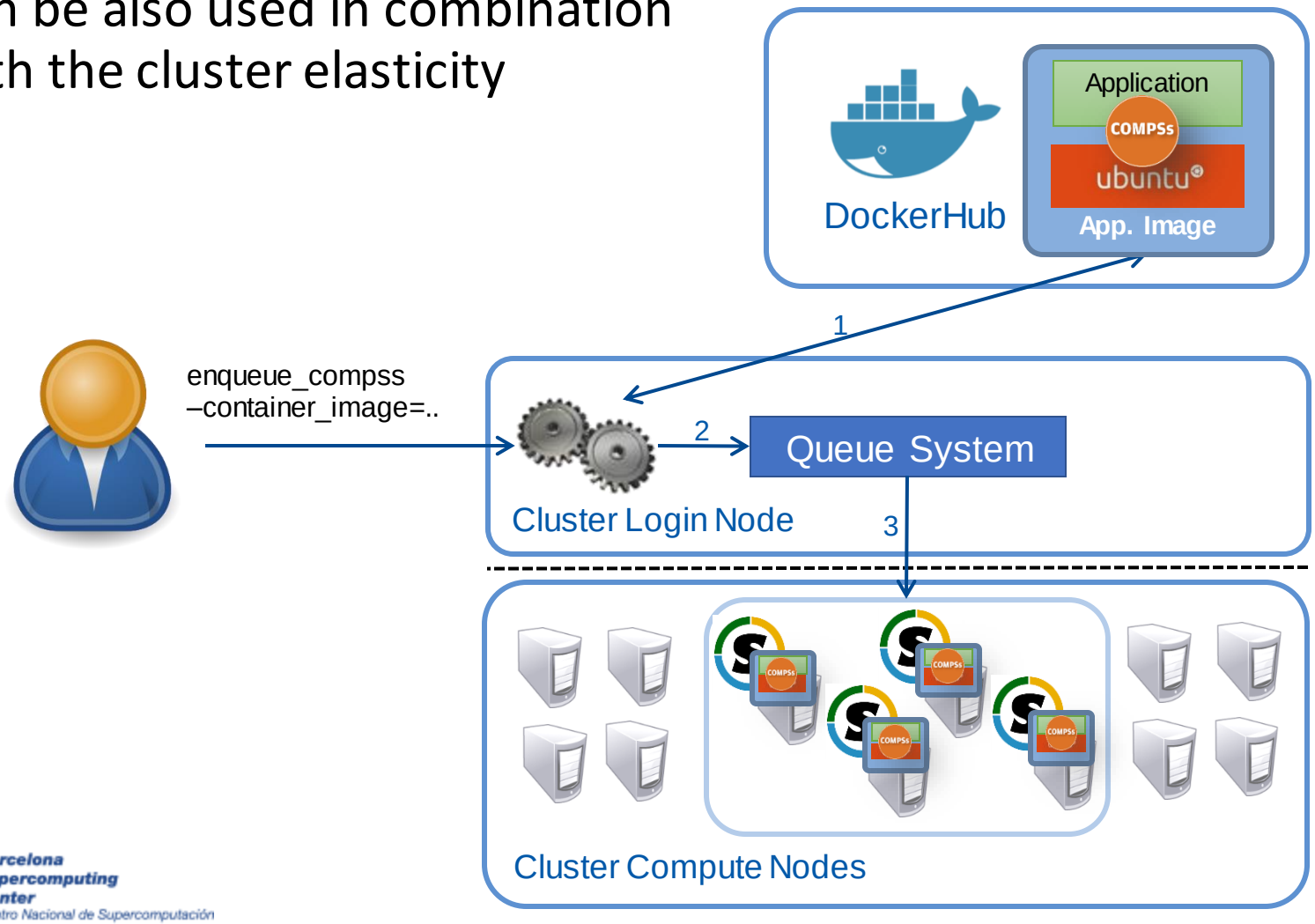
# Elasticity@Docker

- Enable elasticity with the Docker connector (runcompss_docker)



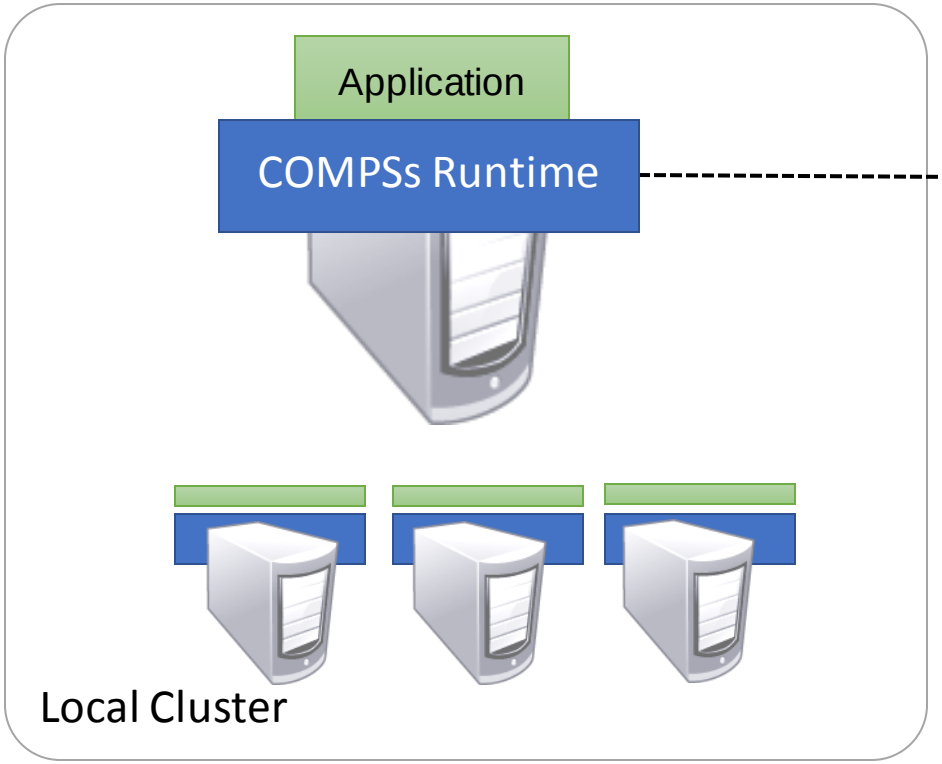
# COMPSs@Singularity

- Execute applications from a container image in HPC cluster
- Can be also used in combination with the cluster elasticity

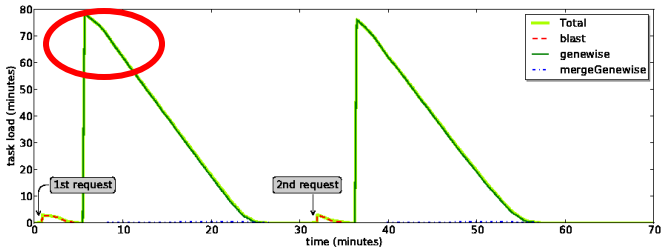


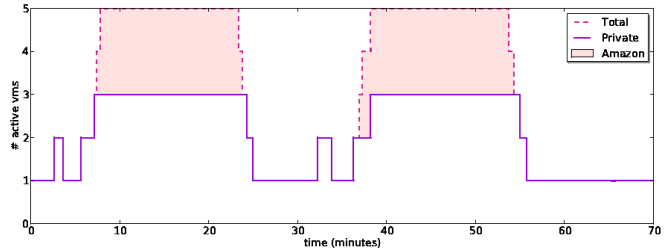
# Cloud Bursting

- Execution of COMPSs applications in Clouds
  - Select de connector to interact Cloud providers  connectors, SSH



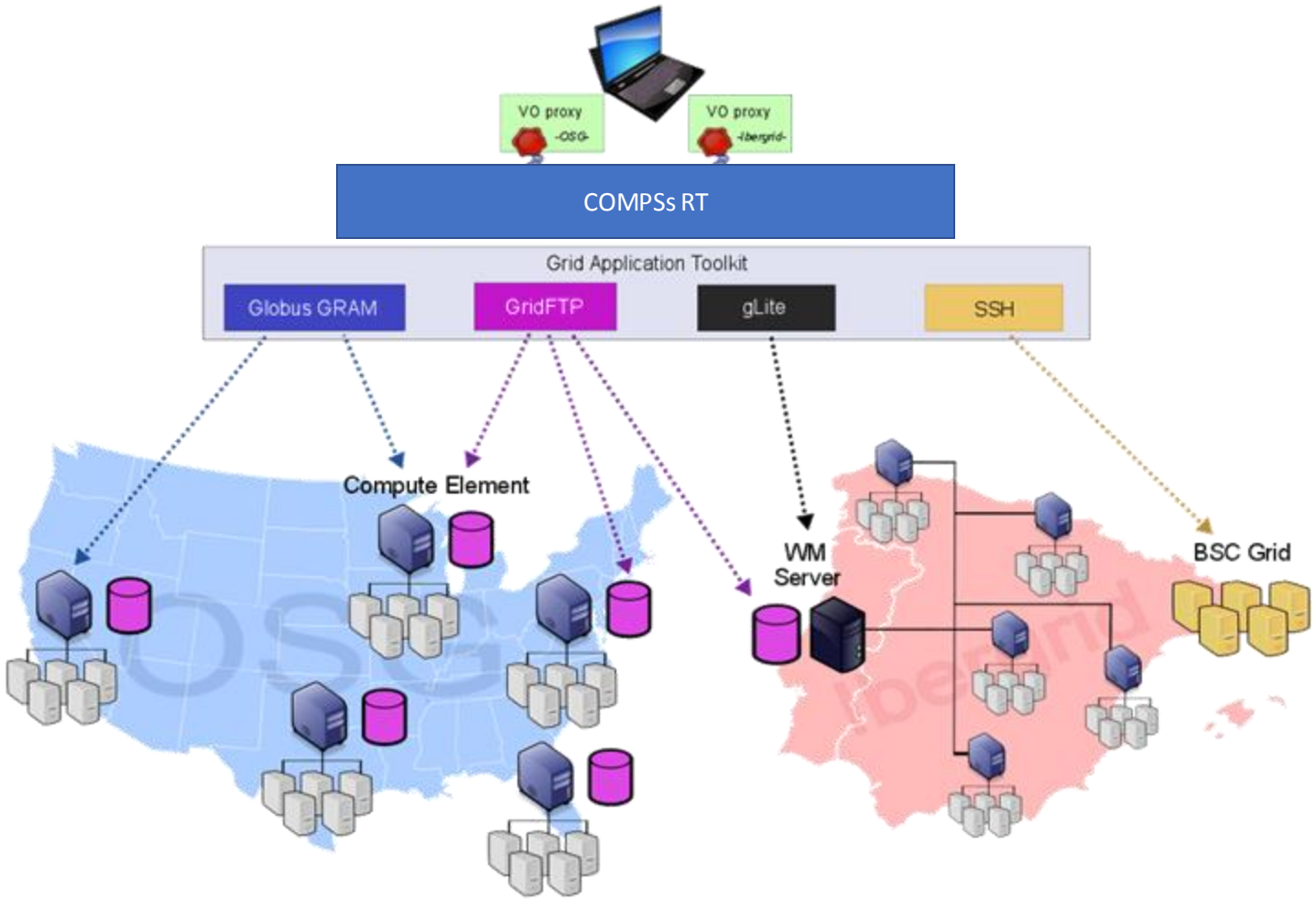





Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# COMPSs@Grid


VO proxy
-OSG-
VO proxy
-Ibergrid-
COMPSs RT
Grid Application Toolkit
Globus GRAM
GridFTP
gLite
SSH
Compute Element
WM
Server
BSC Grid


Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
BSC
EXCELENCIA
SEVERO
OCHOA
THANK YOU!
support-compss@bsc.es
www.bsc.es