Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Programming Distributed Computing Platforms with COMPSs

Workflows & Distributed Computing Group

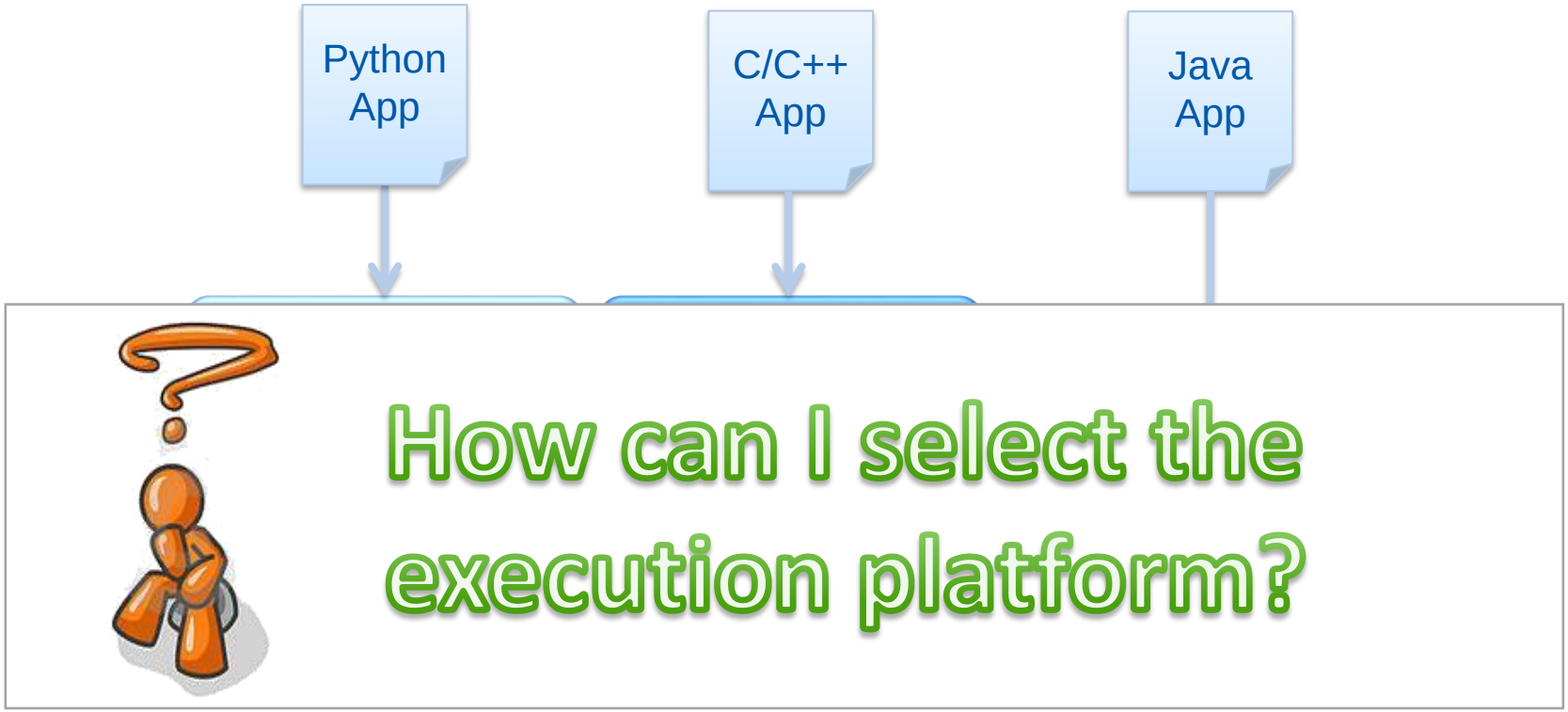
24-25/01/2023

Barcelona and On-line

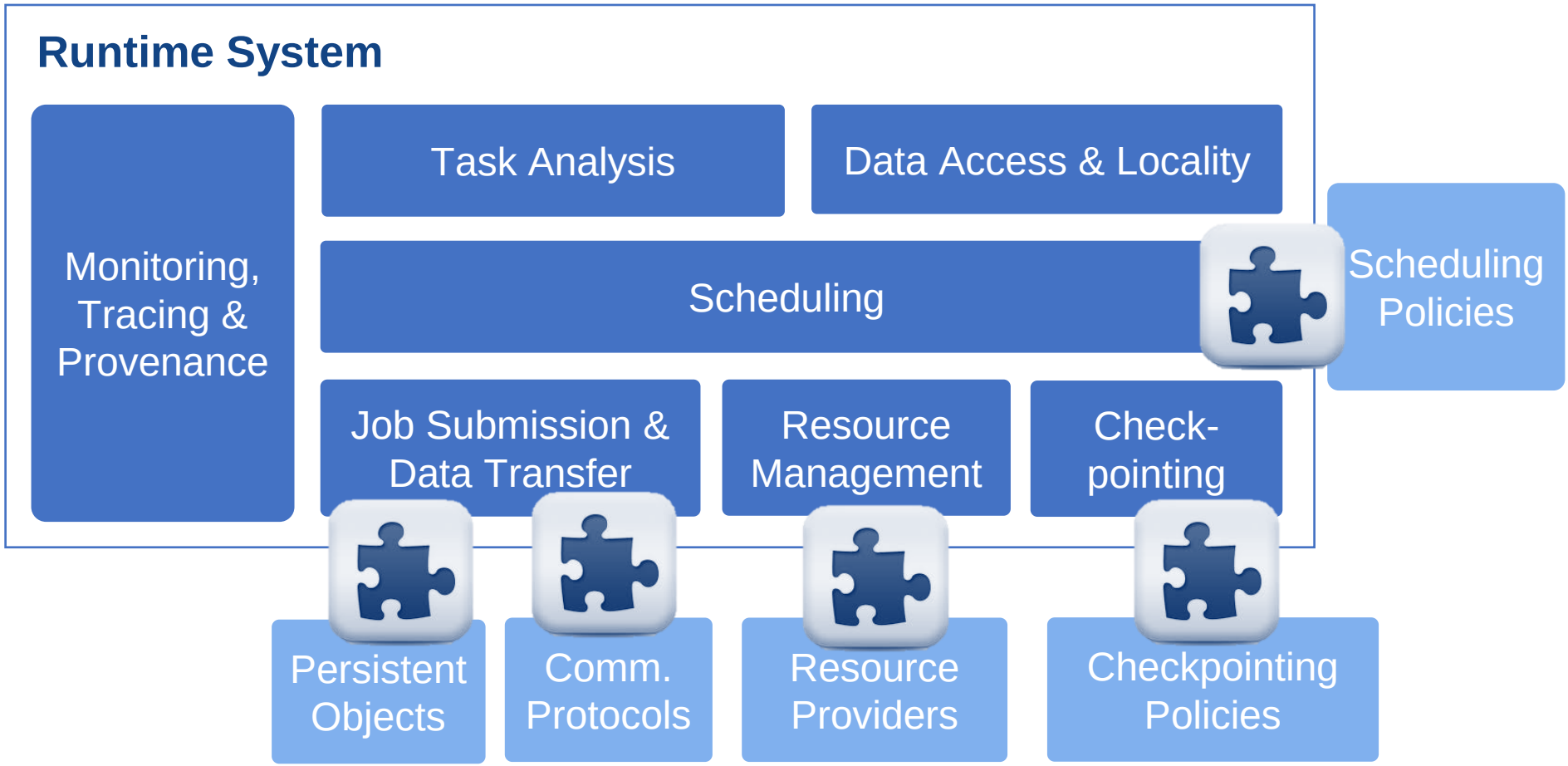
COMPSs Execution Environments

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

compss run 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

Checkpointers

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

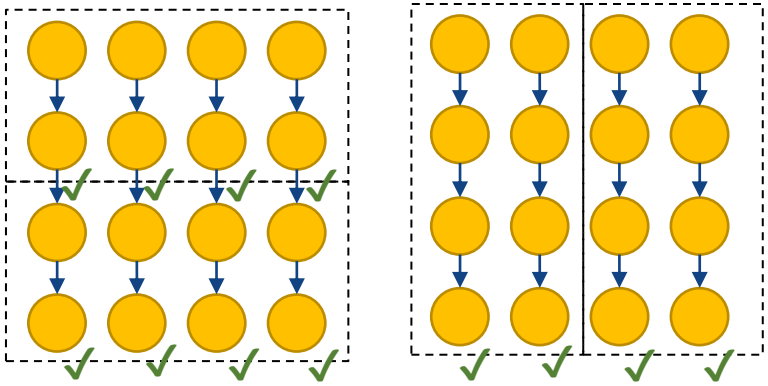
Schedulers

- Specify how to tasks are scheduled
  - Scheduling behaviour:
    - Order strict: Task not scheduled until previous is scheduled
    - Lookahead: Schedule tasks considering all dependency ready
    - Full graph: Schedule tasks considering all generated tasks
  - Task sorting:
    - FIFO, LIFO, Successors, Locality, Constraints
- Usage: ***--scheduler*** flag with *compss run*

Scheduler	Comment
es.bsc.compss.scheduler.**orderstrict.fifo.FifoTS**	Shared disk and Homogenous tasks
es.bsc.compss.scheduler.**lookahead.fifo.FifoTS** es.bsc.compss.scheduler.**lookahead.lifo.LifoTS**	Shared Disks
es.bsc.compss.scheduler.**lookahead.locality.LocalityTS**	Default Scheduler
es.bsc.compss.scheduler.**lookahead.successors.locality.LocalityTS**	Default for local disk on SCs
es.bsc.compss.scheduler.**lookahead.successors.fifo.FifoTS** es.bsc.compss.scheduler.**lookahead.successors.lifo.LifoTS**	Default for shared disk on SCs
es.bsc.compss.scheduler.**lookahead.successors.constraintsfifo.ConstraintsFifoTS**	Computing units
es.bsc.compss.scheduler.**fullgraph.multiobjective.MOScheduler**	Based on a multi-objective function.

Checkpointing

- Mechanism to recover from failures
  - Allows the workflow re-execution avoiding the re-execution of finished tasks
- Asynchronous but with Overhead
  - Save tasks results in a persistent storage
  - Trade-off between performance and time to recover
  - Establishing the right checkpoint granularity is important
- 3 mechanisms for automatic checkpointing
  - **Time:** periodically, COMPSs saves the last version produced for every value
  - **Finished tasks** : after the completion of X tasks, COMPSs saves the last version produced for every value
  - **Instantiation task groups**: Defines groups of tasks, COMPSs saves those data versions that are a final result of the group
- Indicated by the developer with API
- Extensible Policies
  - customize group creations



Activating Checkpointing

- Flag to indicate the checkpointing policy
  - PeriodicTime

```
--checkpoint=es.bsc.compss.checkpoint.policies.CheckpointPolicyPeriodicTime
--checkpoint_params=period.time:1s
```

  - Finished Tasks

```
--checkpoint=es.bsc.compss.checkpoint.policies.CheckpointPolicyFinishedTasks
--checkpoint_params=finished.tasks:3
```

  - Instantiation Groups

```
--checkpoint=es.bsc.compss.checkpoint.policies.CheckpointPolicyInstantiatedGroup
--checkpoint_params=instantiated.group:3
```

- Application developers can also request COMPSs to establish a checkpoint in the application with an API call
  - The runtime will save the version of each data value corresponding to that point of the execution
    - Java `COMPSs.snapshot()`
    - Python `compss_snapshot()`

Automatic Workflow Provenance Recording

- Record details of your COMPSs (and dislib) workflow executions
  - Ensure **reproducibility** and **replicability** of the results
  - **RO-Crate** workflow profile format used (simple and interoperable)

- Usage: ***-p*** or ***--provenance*** flag with *compss run*
  - **Input:** Simple YAML file to describe the application and its authors
  - **Output:** Resulting package with: Application source files, Graph image, Application profiling, RO-Crate metadata file
  - RO-Crate metadata includes:
    - References to detected inputs and outputs used by the workflow (and their details), but **files are not included** (avoid moving large datasets)
    - Application details, COMPSs version used, hostname, …
- More info:

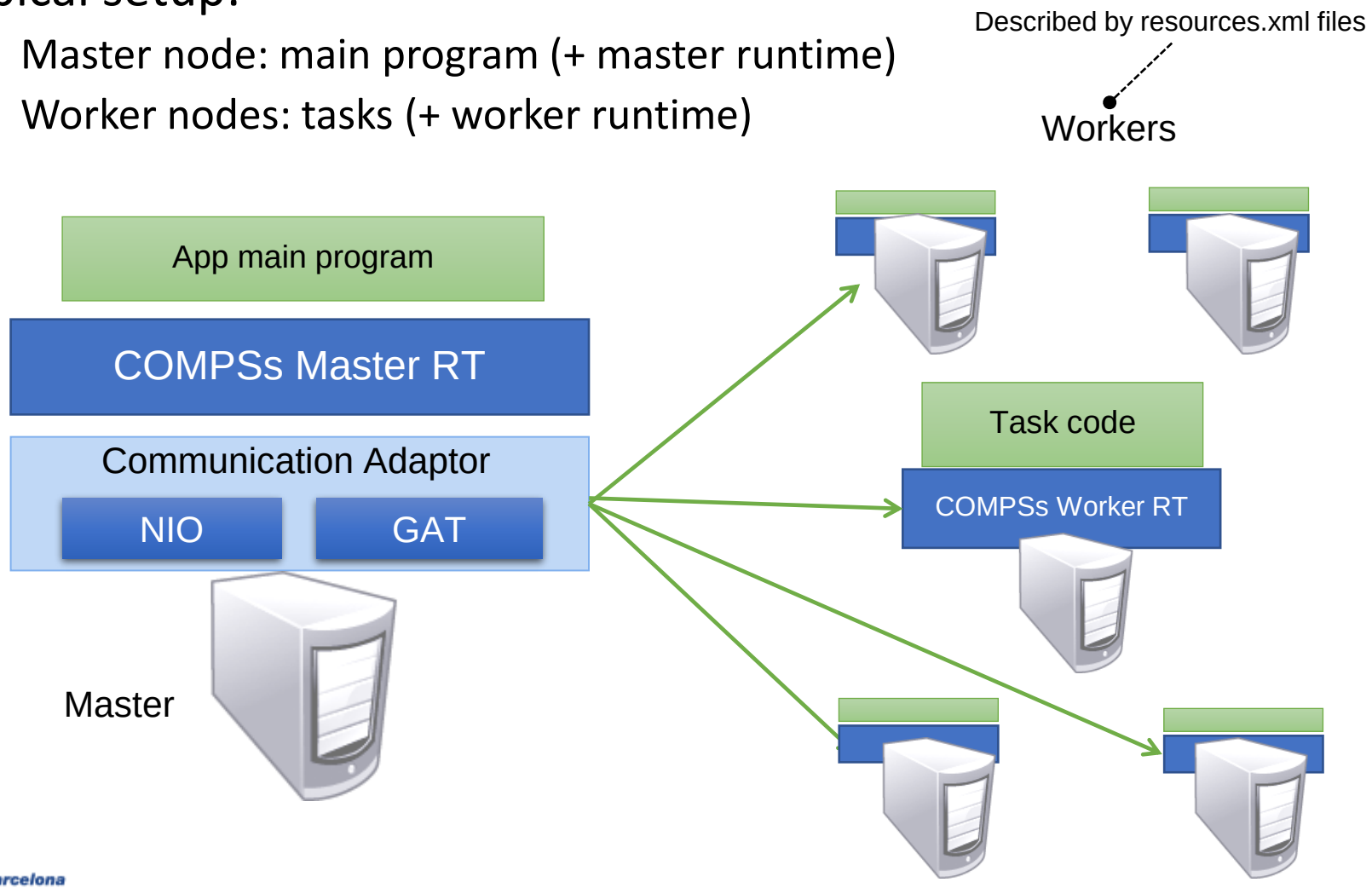
  https://compss-doc.readthedocs.io/en/stable/Sections/05_Tools/04_Data_Provenance.html

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

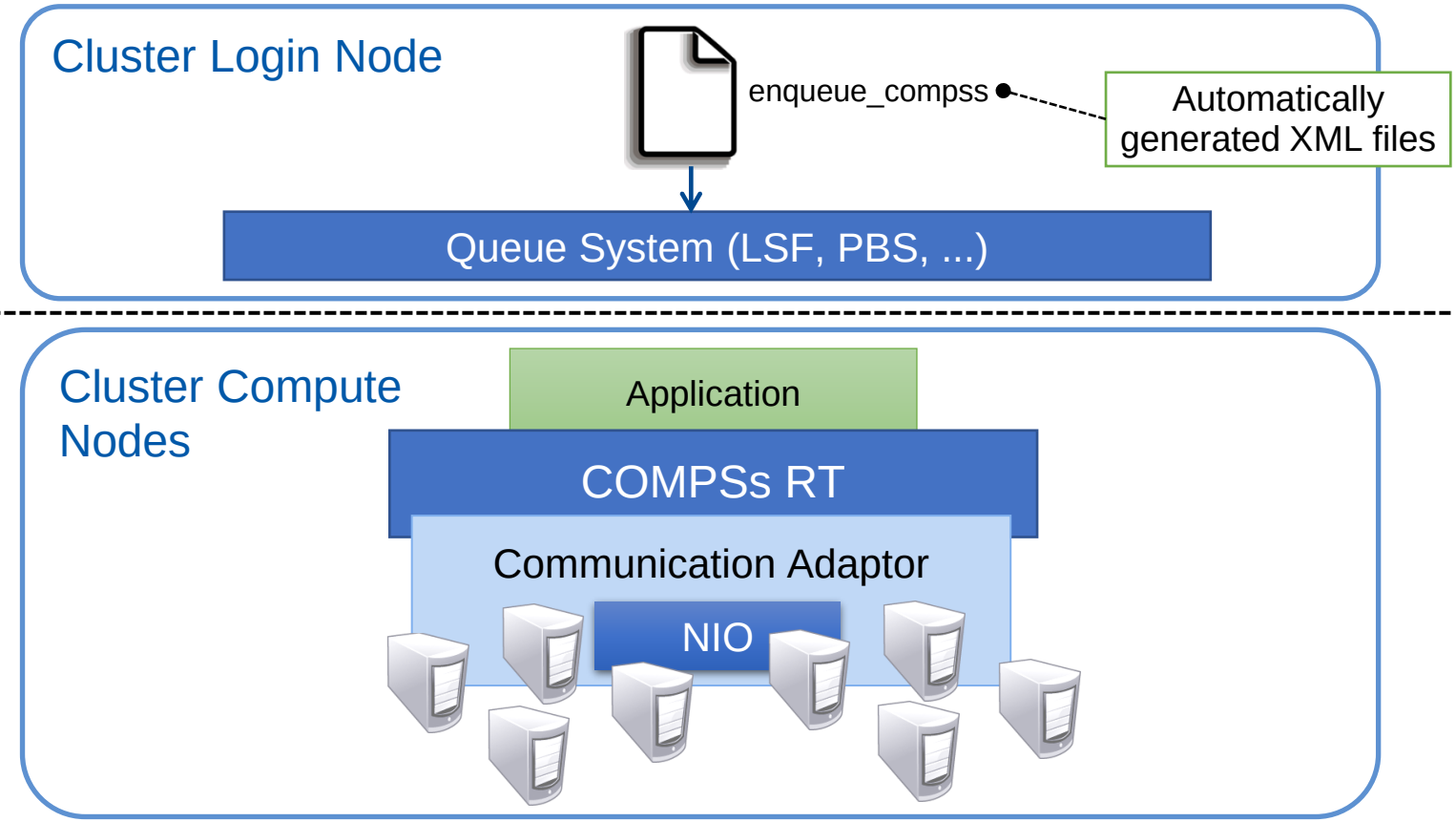
- Project.xml

```
<?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

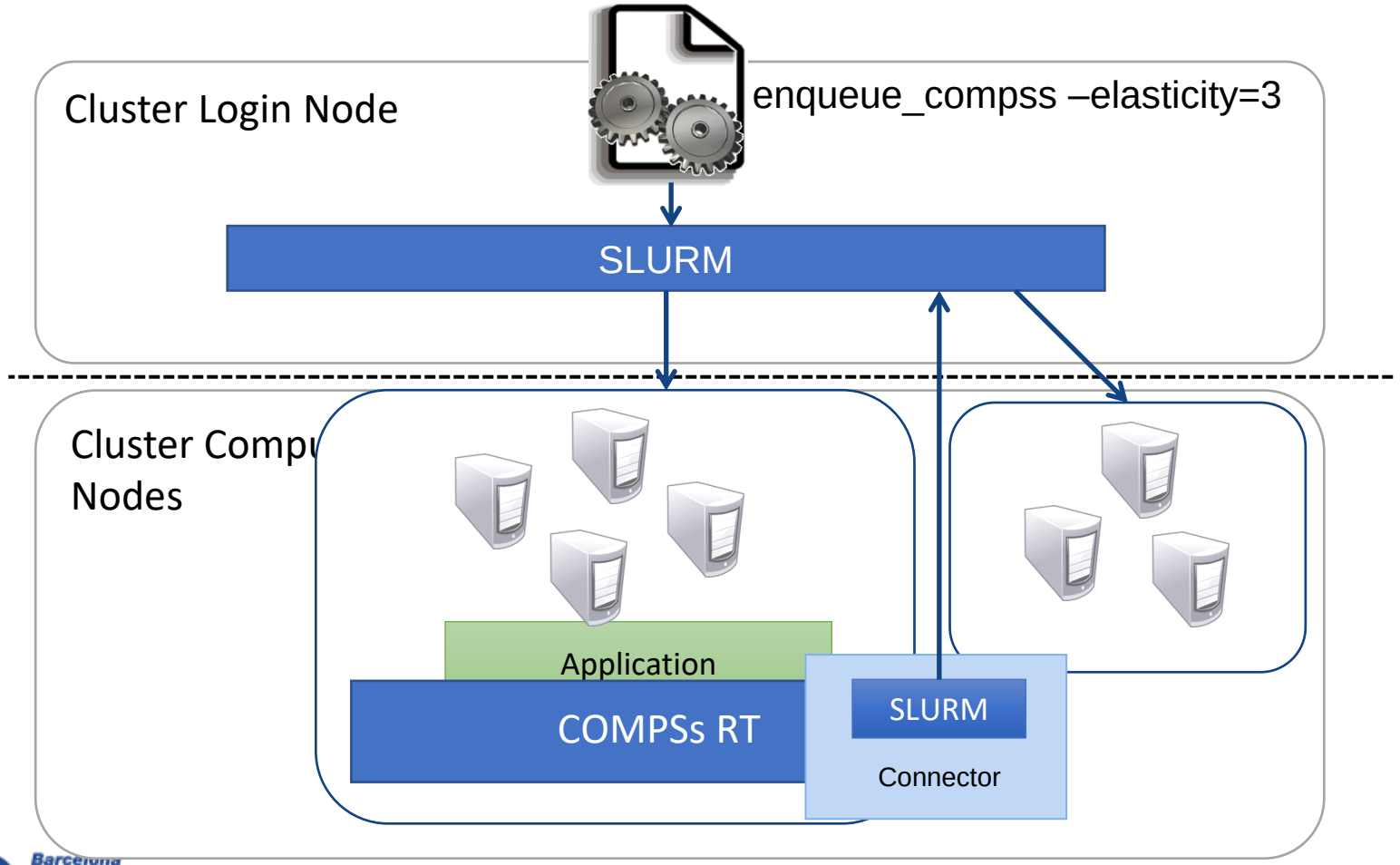
- Execution divided in two phases
  - Launch scripts queue a whole COMPSs app execution
  - Actual execution starts when reservation is obtained



COMPSs@MN

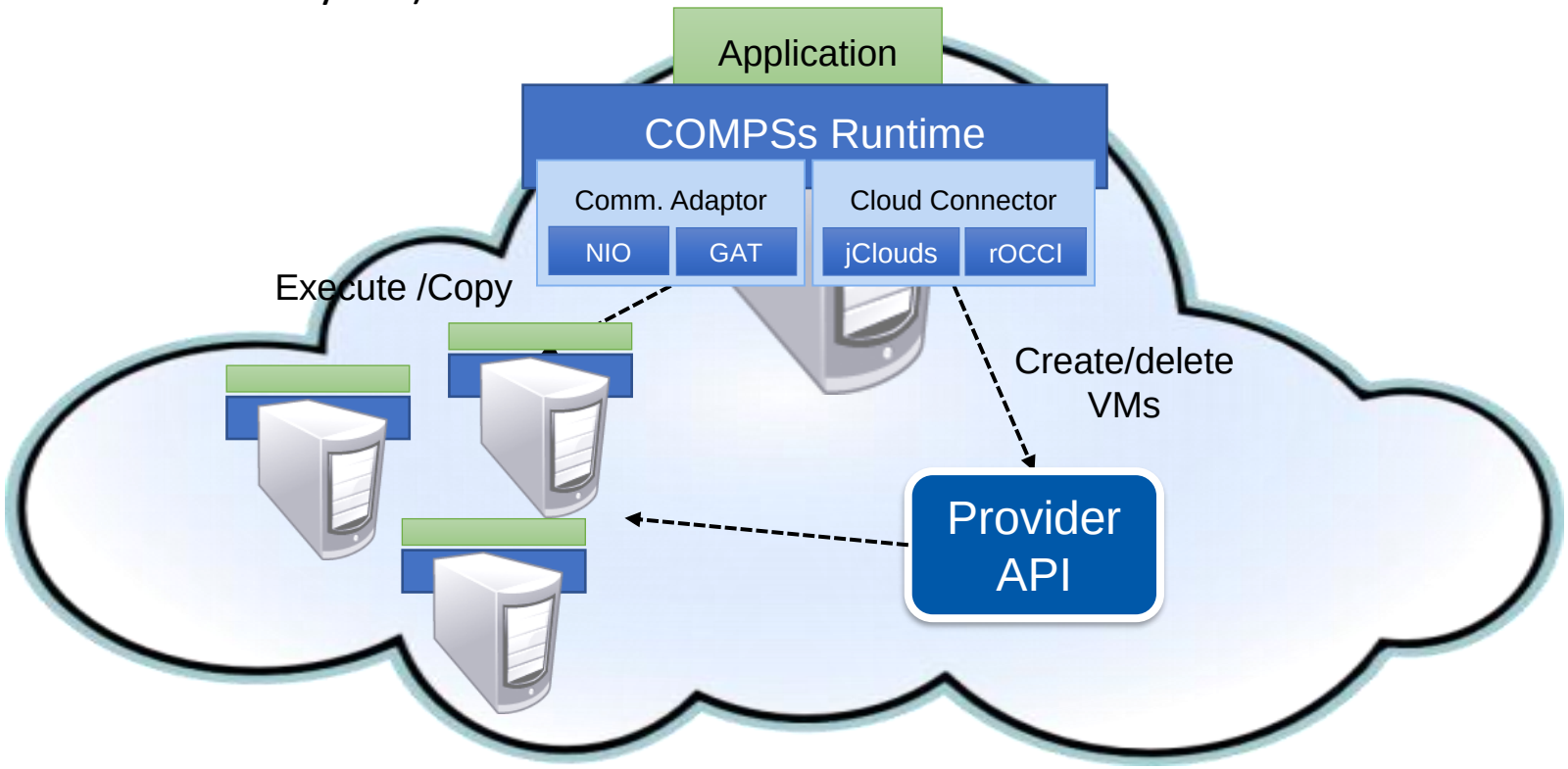
Elasticity@Clusters with SLURM Connector

- Enable the SLURM connector at submission time



COMPSs@Cloud

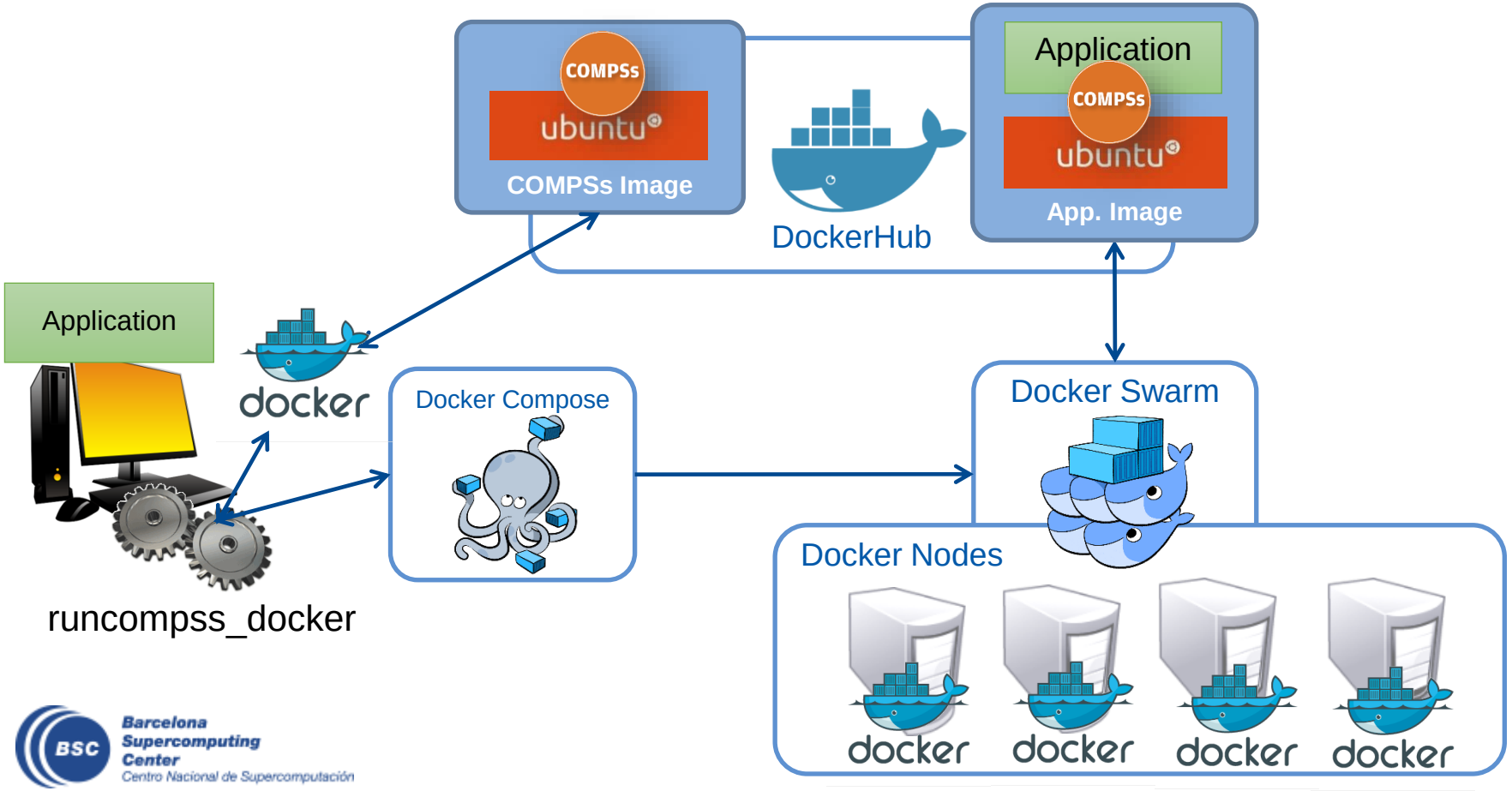
- Execution of COMPSs applications in Clouds
  - Select the connector to interact with the Cloud provider
  - Adaptor to communicate VMs (NIO if provider supports firewall management, GAT if only ssh)



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

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



COMPSs@Singularity

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