Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Programming Distributed Computing Platforms with COMPSs

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

Workflows & Distributed Computing Group

30/01/2018

Barcelona

Outline

- Roundtable (9:30 – 10:00): Presentation and background of participants
  - Session 1 (10:00 – 10:15): Motivation and Introduction to COMPSs
  - Session 2 (10:15-11:00): Python Syntax
- Coffee break (11:00 – 11:30)
  - Session 3 (11:30 – 13:00): Python Hands-on
- Lunch break (13:00-14:00)
  - Session 4 (14:00 -14:30): Java Syntax
  - Session 5 (14:30 -15:30): Java Hands-on
- Coffee break (15:30 – 16:00)
  - Session 6 (16:00 -16:30): COMPSs execution environment
  - Session 7 (16:30 -17:30) Cluster Hands-on (MareNostrum)
- COMPSs Installation & Final Notes
- SLIDES
  - http://compss.bsc.es/releases/tutorials/tutorial-PATC_2018/

Introduction

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Motivation

- New complex architectures constantly emerging
  - With their own way of programming them
    - Fine grain: e.g. APIs to run with GPUs, NVMs (Non-Volatile Memories)
    - Coarse grain: e.g. APIs to deploy in Clouds
  - **Difficult** for programmers
    - Higher learning curve / Time To Market (TTM)
    - What about non computer scientists???
  - **Difficult** to understand what is going on during execution
    - Was it fast? Could it be even faster? Am I paying more than I should? (**Efficiency**)
  - Tune your application for each architecture (or cluster)
    - E.g. partitioning data among nodes

Motivation

- Create tools that make user's life **easier**
  - Intermediate layer: let the difficult parts to those tools
    - Act on behalf of the user
    - Distributing the work through resources
    - Dealing with architecture specifics
    - Automatically improving performance
  - Tools for visualization
    - Monitoring
    - Performance analysis

BSC vision on programming models

Applications

PM: High-level, clean, abstract interface

Power to the runtime

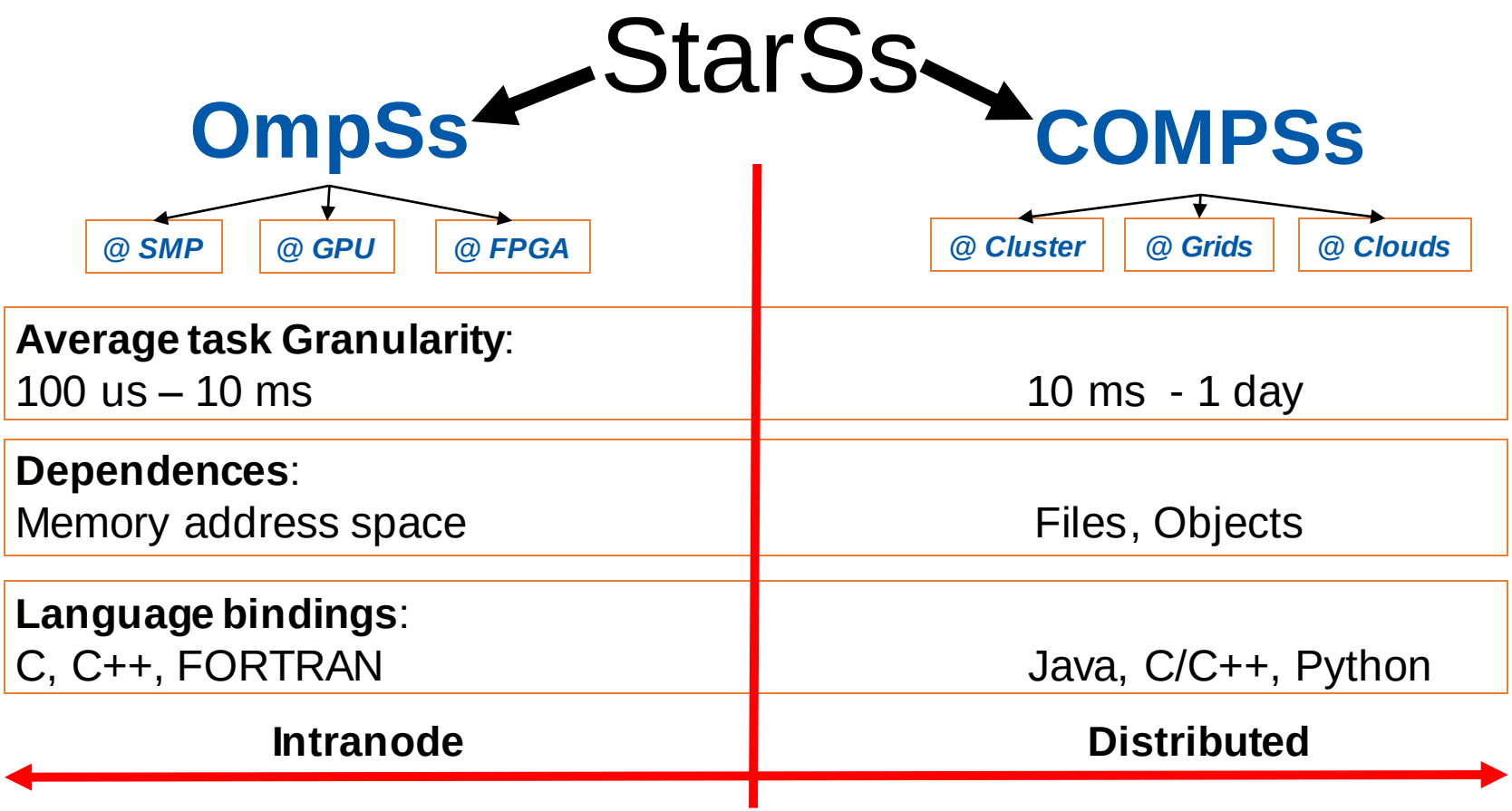
API

Program logic independent of computing platform

General purpose
Task based
Single address space

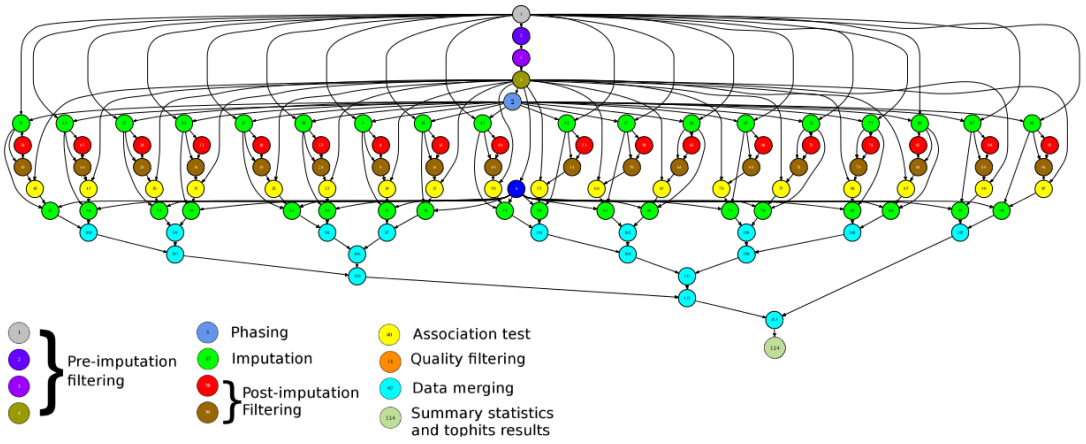
Intelligent runtime, parallelization, distribution, interoperability

BSC vision on programming models

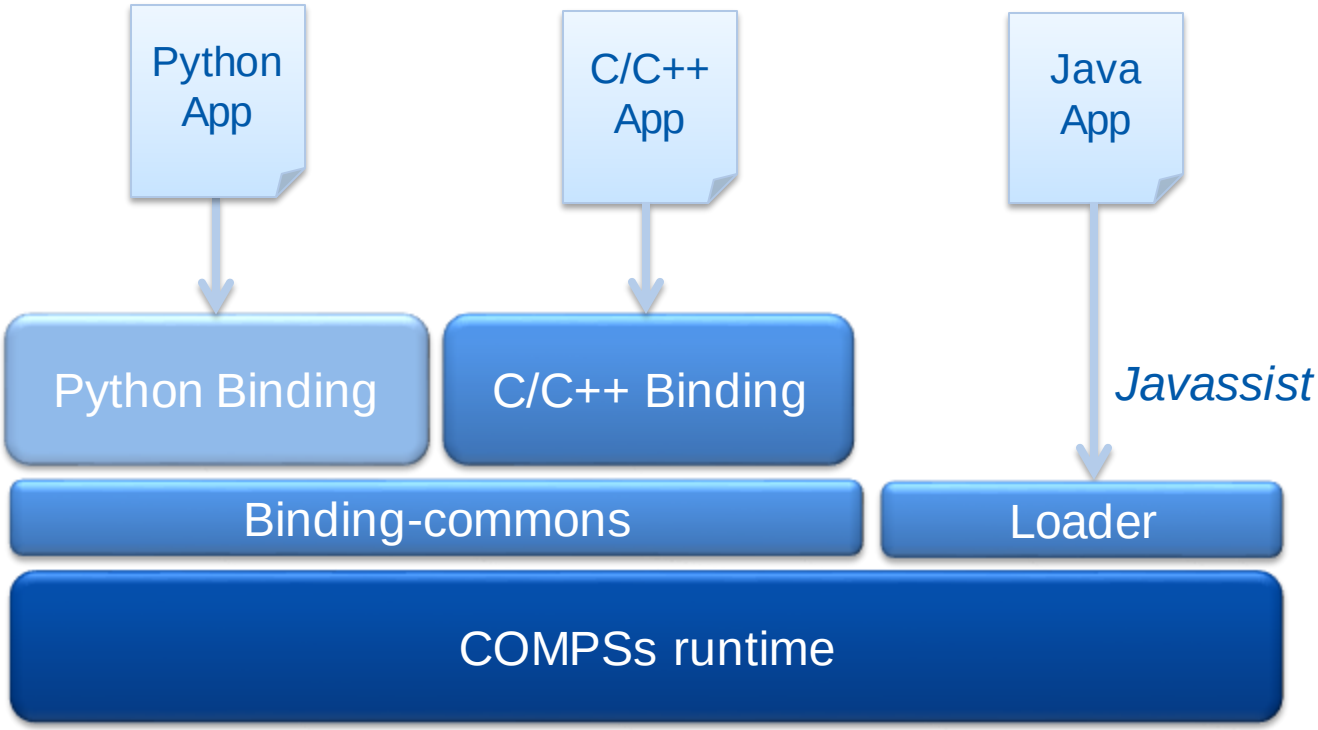


Programming with COMPSs

- 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
- Exploitation of parallelism
  - … and of distant parallelism
- Agnostic of computing platform
  - Enabled by the runtime for clusters, clouds and grids



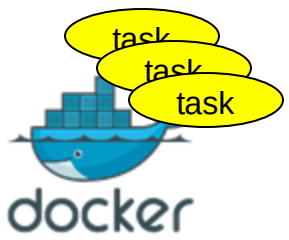
COMPSs Architecture



Grid

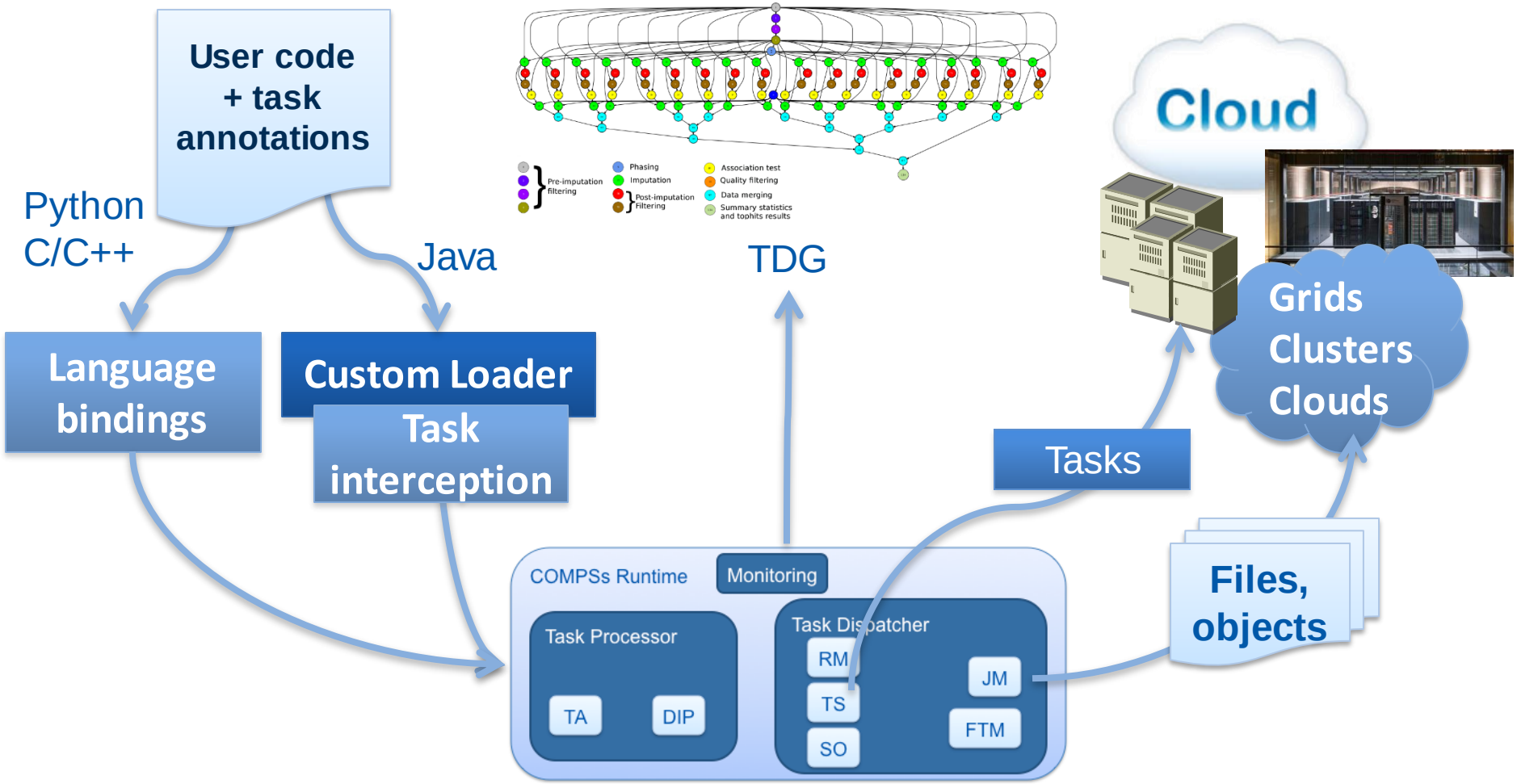
Cluster

Cloud



Containers

COMPSs Runtime



Python Syntax (PyCOMPSs)

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. *

- 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 (>127,000 as of January 2018) **

* From python.org ** From pypi.python.org

PyCOMPSs - Virtual appliance

- Virtual appliance ready to play with PyCOMPSs/COMPSs
- Available at:

  http://compss.bsc.es

  - Downloads
    - Virtual appliances

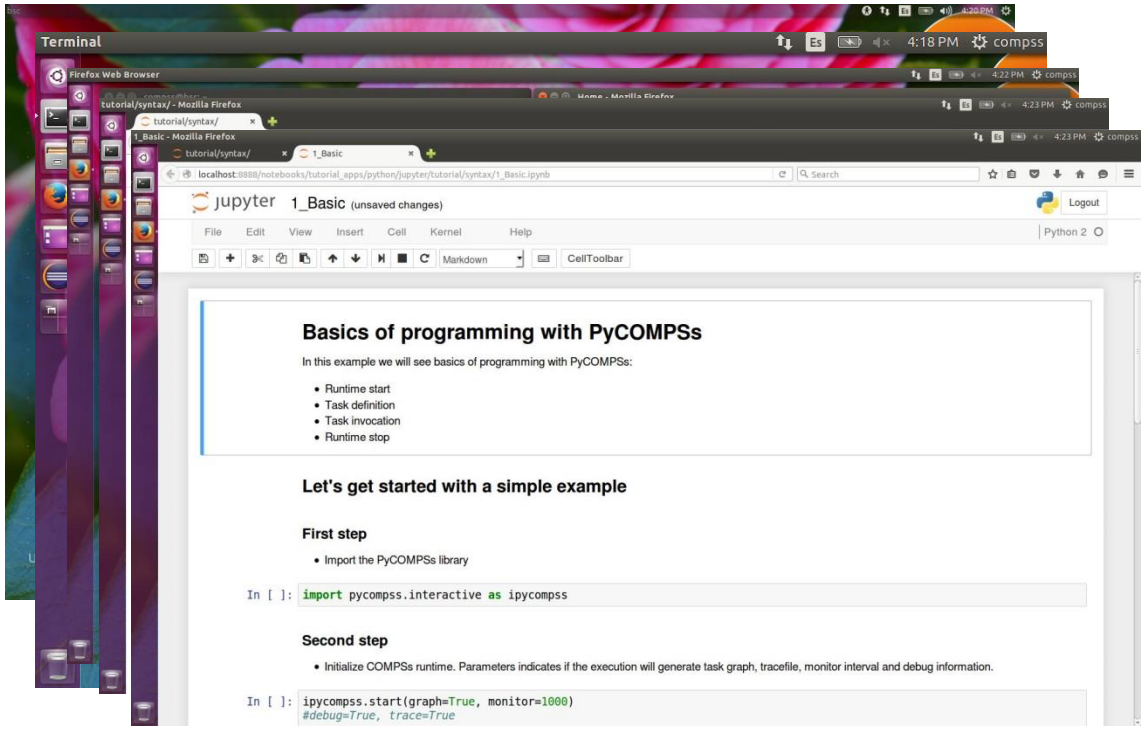
Python Syntax (Jupyter-notebook)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

PyCOMPSs and Jupyter-Notebook

- Start the Virtual Machine
  - User: compss
  - Password: compss2017
- Open a console
- Start jupyter-notebook
  - $ jupyter-notebook
- Look for the tutorial notebooks:

/tutorial_apps/python/jupyter/tutorial/syntax



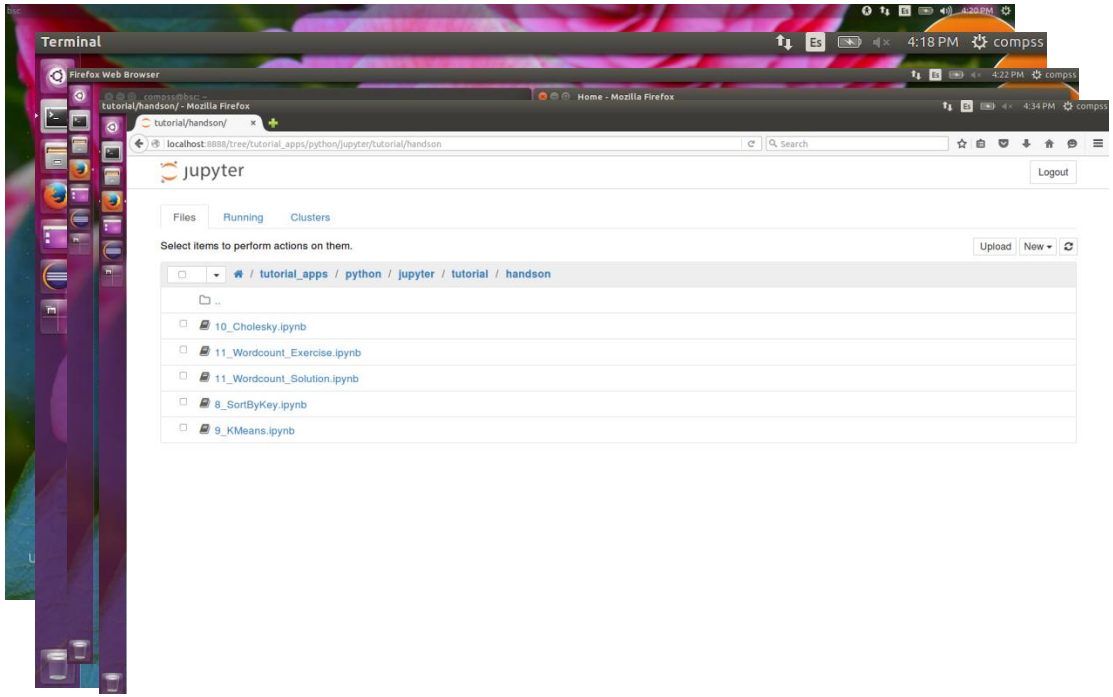
Python Hands-on (Jupyter-notebook)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

PyCOMPSs and Jupyter-Notebook

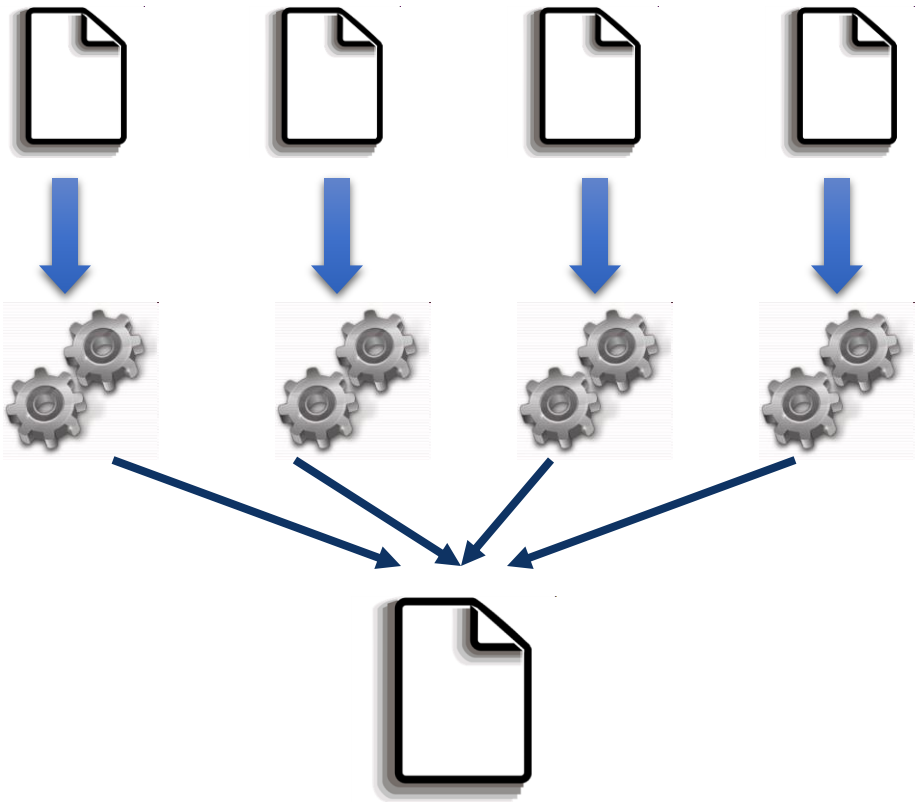
- Start the Virtual Machine
  - User: compss
  - Password: compss2017
- Open a console
- Start jupyter-notebook
  - $ jupyter-notebook
- Look for the hands on notebooks:
  - SortByKey demo
  - Kmeans demo
  - Cholesky demo
  - Wordcount Exercise

/tutorial_apps/python/jupyter/tutorial/handson



Wordcount

- Counting words of a set of documents
- Parallelization
  - Phase 1: Count words of a set of documents
  - Phase 2 : Merge results



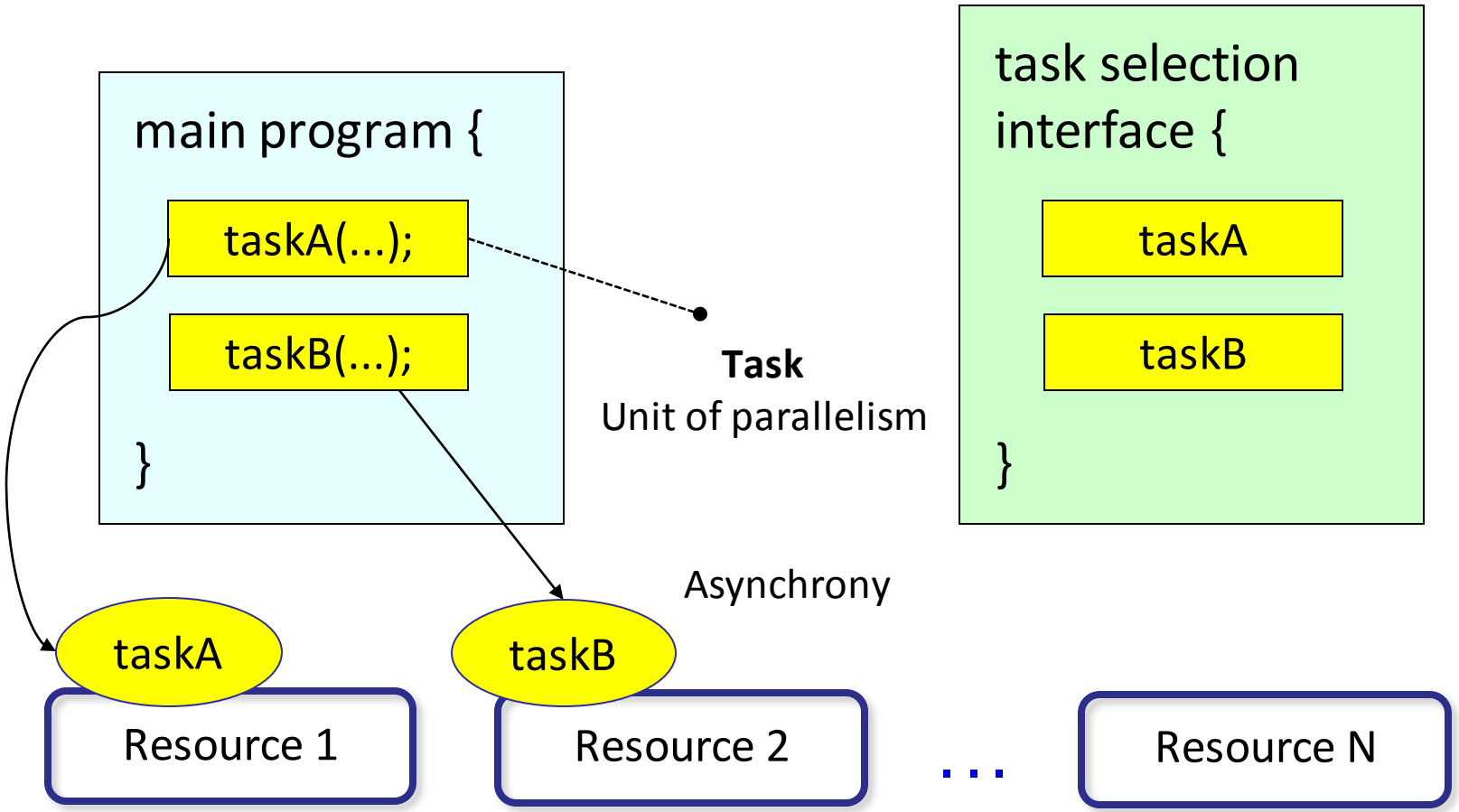
Java Syntax

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Programming Steps

1. Identify tasks

2. Select tasks



Task Selection Interface

```
public interface SampleItf {
   @Constraints(computingUnits = "1", memorySize = "0.5f")
   @Method(declaringClass = "servicess.Example")
    void myMethod(
        @Parameter(direction = INOUT)
        Reply r
     );

   @Service(namespace = "http://servicess.es/example",
            name = "SampleService",
            port = "SamplePort")
    Reply myServiceOp(
        @Parameter(direction = IN)
        Query q
     );
}
```

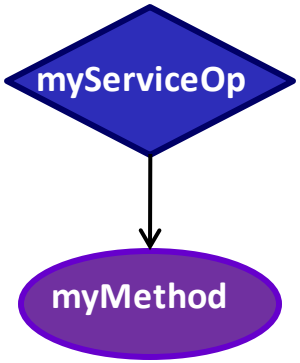
Main program

```
public class App {

   public static void main(String[] args) {
       Query query = new Query(…);
       Reply reply = myServiceOp(query);

       myMethod(reply);

       reply.printToLog();
   }
}
```

Service task call

Method task call

Synchronization



Versioning

```
@Constraints(computingUnits = “1”, memorySize = “0.5f”)
@Method(declaringClass = “example.Sequential”)
@Method(declaringClass = “example.Threading”,
  constraints = @Constraints(computingUnits = "2"))
void myMethod(
      @Parameter(direction = INOUT)
      Reply r
    );
}
```

Java example

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Sample Application

- Main Program

```
public static void main(String[] args) {
   String counter1 = args[0], counter2 = args[1], counter3 = args[2];

   initializeCounters(counter1, counter2, counter3);

   for (i = 0; i < 3; i++) {

      increment(counter1);
      increment(counter2);
      increment(counter3);

   }
}
```

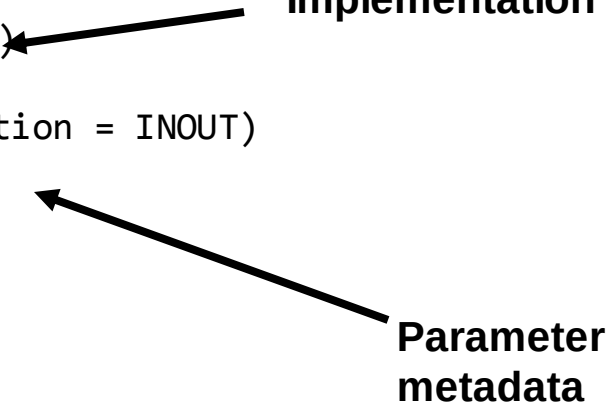
- Task Method

```
public static void increment(String counterFile) {
      int value = readCounter(counterFile);
      value++;
      writeCounter(counterFile, value);
}
```

Sample Application (Interface)

- Task Annotation Interface

```
public interface SimpleItf {

        @Method(declaringClass  = “SimpleImpl")
        void increment(
                @Parameter(type = FILE, direction = INOUT)
                String counterFile
        );

}
```



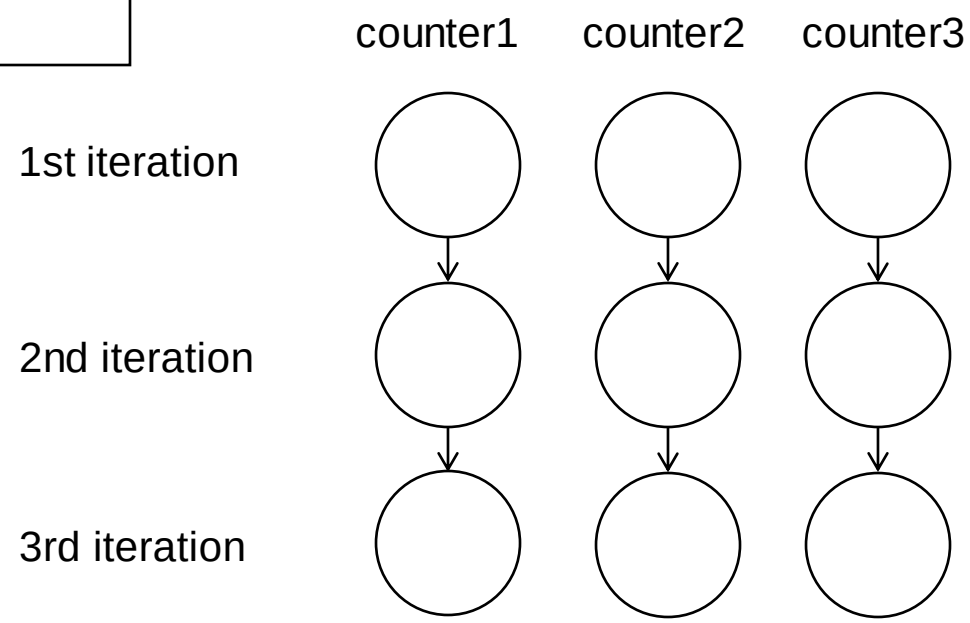
Sample Application (Main Program)

- Main program NO CHANGES!

```
public static void main(String[] args) {
   String counter1 = args[0], counter2 = args[1], counter3 = args[2];

   initializeCounters(counter1, counter2, counter3);

   for (i = 0; i < 3; i++) {

      increment(counter1);
      increment(counter2);
      increment(counter3);

   }
}
```

Programming Model: Task Graph

```
for (i = 0; i < 3; i++) {

      increment(counter1);
      increment(counter2);
      increment(counter3);

}
```



Integrating Binaries and MPI applications with COMPSs in Java

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Integrating Binaries

```
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

```
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 */
  }
}
```

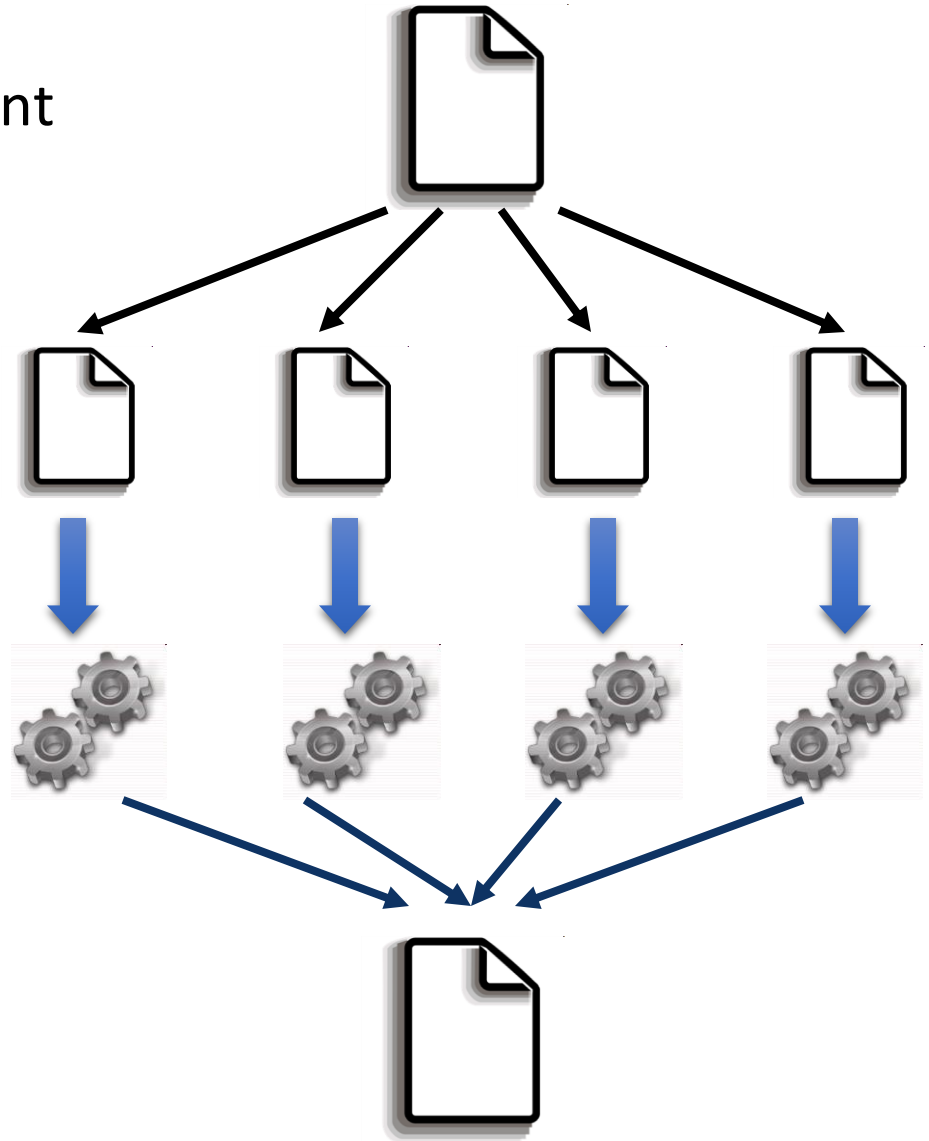
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Java Hands-on

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Word count

- Counting words of a document
- Parallelization
  - Split documents in blocks
  - Count words of Blocks
  - Merge results



Java Hands On: Exercise

- Complete the Word Count parallelization with COMPSs
  - Level 0: No Java background
    - Look the implementation (wordcount project)
  - Level 1: Basic Java background
    - Define methods in the interface (wordcount_sequential)
  - Level 2: Java background
    - Define methods in the interface and complete the part of the main code with helper methods (wordcount_blanks)

Java Hands On: Exercise Solution

- Main Code

```
private static void computeWordCount() {
        HashMap<String, Integer> result  = new HashMap<String, Integer>();
        int start = 0;
        for (int i = 0; i < NUM_BLOCKS; ++i) {
          HashMap<String, Integer> partialResult = wordCountBlock(DATA_FILE, start, BLOCK_SIZE);
          start = start + BLOCK_SIZE;
          result = mergeResults(result, partialResult);
        }
        System.out.println("[LOG] Counted Words is : " + result.keySet().size());
}
```

- Interface

```
public interface WordcountItf {
        @Method(declaringClass  = "wordcount.uniqueFile.Wordcount")
        public HashMap<String, Integer> mergeResults(
          @Parameter HashMap<String, Integer> m1,
          @Parameter HashMap<String, Integer> m2
        );

        @Method(declaringClass  = "wordcount.uniqueFile.Wordcount")
        HashMap<String, Integer> wordCountBlock(
          @Parameter(type = Type.FILE, direction = Direction.IN) String filePath,
          @Parameter int start,
          @Parameter int bsize
        );
}
```

Compilation and Simple Execution

- Compilation (Eclipse IDE)
  - Package Explorer -> Project (wordcount) -> Export… (Solution)
- Use runcompss command to run the application
  - runcompss [options] < FQDN app. classname> <application args>

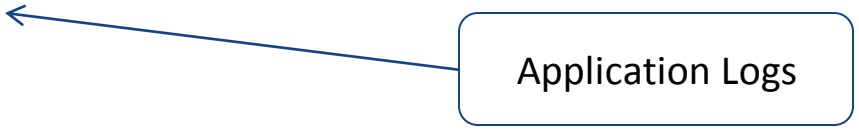
- ***Exercise:*** Simple word count execution
  - Usage:

    wordcount.uniqueFile.Wordcount <data_file> <block_size>


```
$compss@bsc:~/> cd ~/tutorial_apps/java/wordcount/jar
$compss@bsc:~/tutorial_apps/java/wordcount/jar/> runcompss wordcount.uniqueFile.Wordcount
    /home/compss/tutorial_apps/java/wordcount/data/file_short.txt 650
```

Java Hands-on: Result

```
$compss@bsc:~/tutorial_apps/java/wordcount/jar/> runcompss wordcount.uniqueFile.Wordcount
    ~/tutorial_apps/java/wordcount/data/file_short.txt 650

Using default location for project file:
    /opt/COMPSs/Runtime/scripts/user/../../configuration/xml/projects/project.xml
Using default location for resources file:
    /opt/COMPSs/Runtime/scripts/user/../../configuration/xml/resources/resources.xml

----------------- Executing wordcount.uniqueFile.Wordcount -------------------------

WARNING: IT Properties file is null. Setting default values
[   API]  -  Deploying COMPSs Runtime v2.2 (build xxxx)
[   API]  -  Starting COMPSs Runtime v2.2 (build xxxx)
DATA_FILE parameter value = /home/compss/tutorial_apps/java/wordcount/data/file_short.txt
BLOCK_SIZE parameter value = 650
[LOG] Computing word count result
[LOG] Counted Words is : 250
[   API]  -  No more tasks for app 1
[   API]  -  Getting Result Files 1
[   API]  -  Execution Finished

------------------------------------------------------------
```



Java Hands-on: Configuration

- Project.xml: /opt/COMPSs/Runtime/configuration/xml/projects/default_project.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Project>
      <MasterNode>
      <ComputeNode  Name="localhost">
                <InstallDir>/opt/COMPSs/</InstallDir>
          <WorkingDir>/tmp/COMPSsWorker</WorkingDir>
       </ComputeNode>
</Project>
```

  - Other optional parameters
    - User, AppDir, LibraryPath

Java Hands-On: Configuration

- Resources.xml: /opt/COMPSs/Runtime/configuration/xml/resources/default_resources.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ResourceList>
    <!--Description for any physical node-->
    <ComputeNode Name="localhost">
      <Processor Name="Main">
                <ComputingUnits>4</Computingnits>
          </Processor>
          <Memory>
          <size>8</size>
      </Memory>
       <Storage>
            <size>50</size>
      </Storage>
            <Adaptors>
          <Adaptor Name="integratedtoolkit.nio.master.NIOAdaptor">
            ...
          <Adaptor Name="integratedtoolkit.gat.master.GATAdaptor">
            ...
            </Adaptors>
    </ComputeNode>
<ResourceList>
```

Affects to application parallelism

Java Hands-On: Monitoring

- The runtime of COMPSs provides real-time monitoring
  - http://localhost:8080/compss-monitor/
  - If not started run as **root**:
    - /etc/init.d/compss-monitor start
- The user can log-in and follow the progress of the executions
  - Running tasks, resources usage, execution time per task, real-time execution graph, etc.
- Activate monitoring with a runcompss flag
  - Setting a monitoring interval
    - runcompss **--monitoring=<int>**
  - With a default monitoring interval
    - runcompss **–m** (or) runcompss **--monitoring**
- ***Exercise:*** run wordcount enabling monitoring


```
$compss@bsc:~/> cd ~/tutorial_apps/java/wordcount/jar
$compss@bsc:~/tutorial_apps/java/wordcount/jar/> runcompss –m wordcount.uniqueFile.Wordcount
    /home/compss/tutorial_apps/java/wordcount/data/file_long.txt 250000
```

Java Hands-on: Debugging

- Different log levels activated as runcompss options
  - runcompss **--log_level**=<level>
    (**off**: for performance | **info**: basic logging | **debug**: detect errors)
  - runcompss **–debug** or runcompss **-d**
- The output/errors of the main code of the application are shown in the console
- Other logging files are stored in:
  - $HOME/.COMPSs/<APP_NAME>_XX
- Inside this folder, the user can check the following:
  - The output/error of a task # N : ***/jobs/jobN.[out|err]***
  - Messages from the COMPSs : ***runtime.log***
  - Task to resources allocation: ***resources.log***
- ***Exercise:*** run wordcount with debugging


```
$compss@bsc:~/> cd ~/tutorial_apps/java/wordcount/jar
$compss@bsc:~/tutorial_apps/java/wordcount/jar/> runcompss –d wordcount.uniqueFile.Wordcount
    /home/compss/workspace_java/wordcount/data/file_short.txt 650
```

Java Hands-on: Graph generation

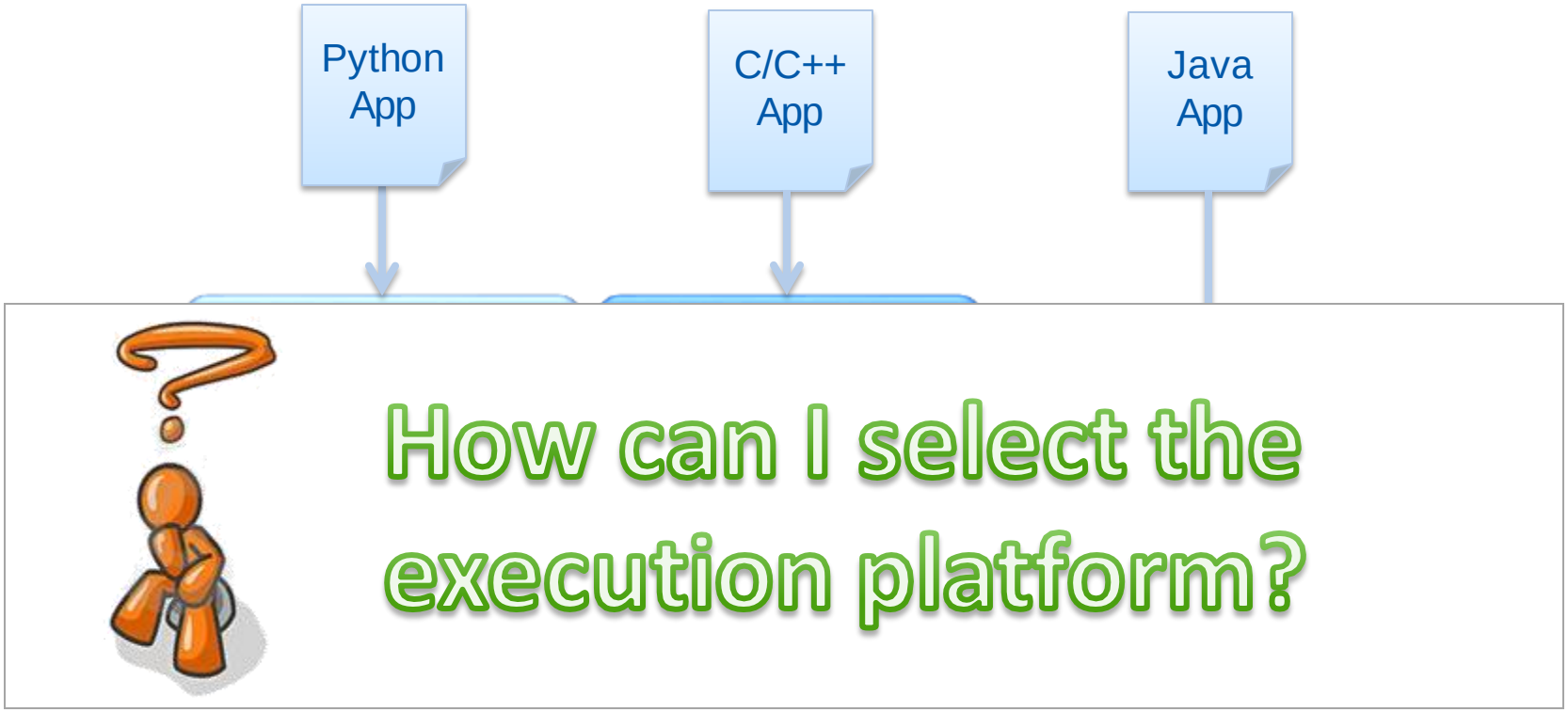
- To generate the graph of an application, it must be run with the monitor or graph flags activated
  - runcompss **-m** (or) runcompss **–graph** (or) runcompss **-g**
- The graph will be stored in:
  - $HOME/.COMPSs/<APP_NAME>_XX/monitor/complete_graph.dot
- To convert the graph to a PDF format use gengraph command
  - Usage: gengraph <dot_file>
- **Exercise:** generate the graph for the wordcount application


```
$compss@bsc:~/> cd ~/tutorial_apps/java/wordcount/jar
$compss@bsc:~/tutorial_apps/java/wordcount/jar/> runcompss –g wordcount.uniqueFile.Wordcount
      /home/compss/tutorial_apps/java/wordcount/data/file_short.txt 650

… application execution …

$compss@bsc:~/tutorial_apps/java/wordcount/jar/> cd ~/.COMPSs/wordcount.uniqueFile.Wordcount_04/monitor
$~/.COMPSs/wordcount.uniqueFile.Wordcount_04/monitor> gengraph complete_graph.dot
Output file: complete_graph.pdf
$~/.COMPSs/wordcount.uniqueFile.Wordcount_04/monitor> evince complete_graph.pdf
```

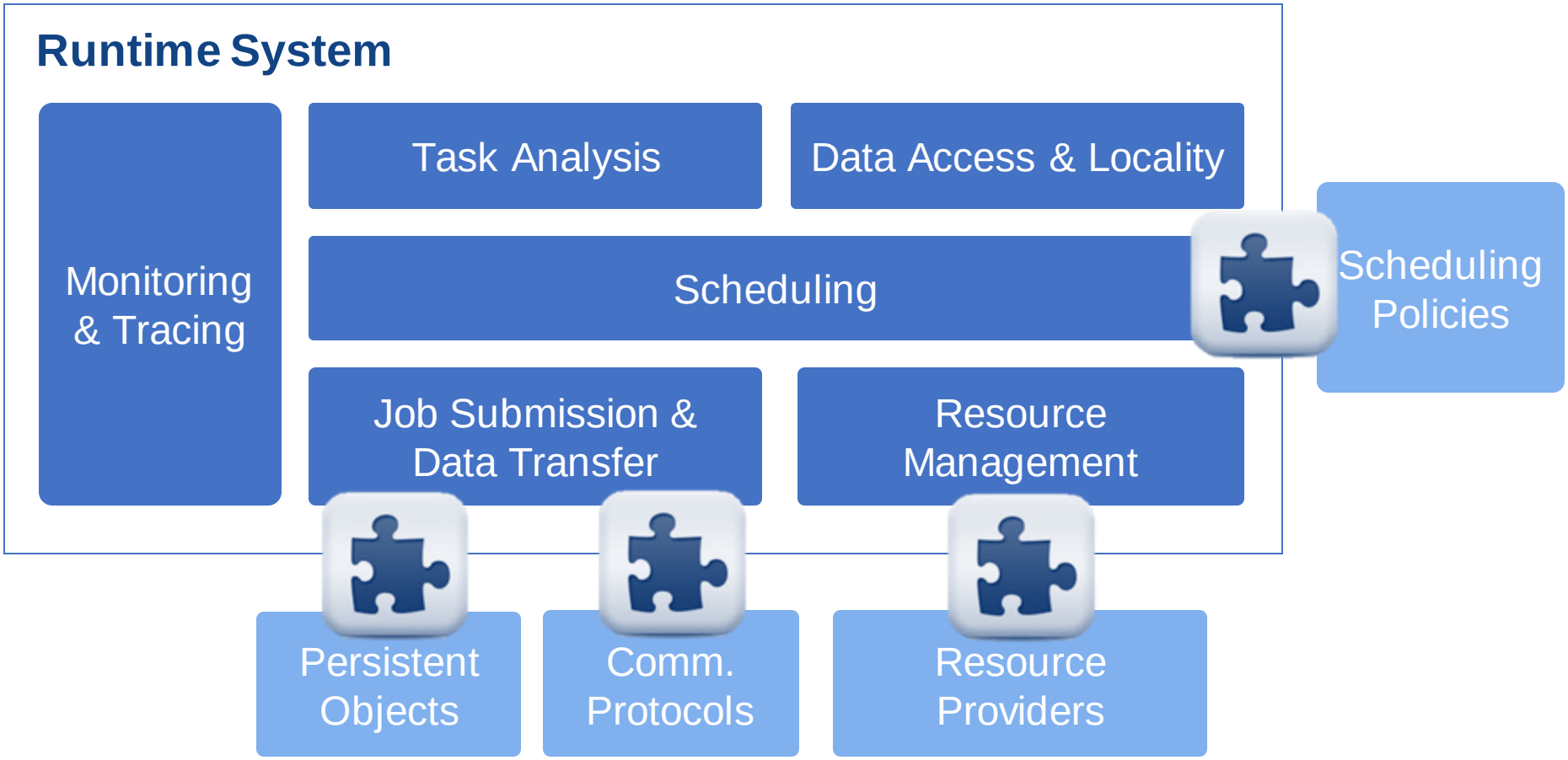
COMPSs Execution Environments

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

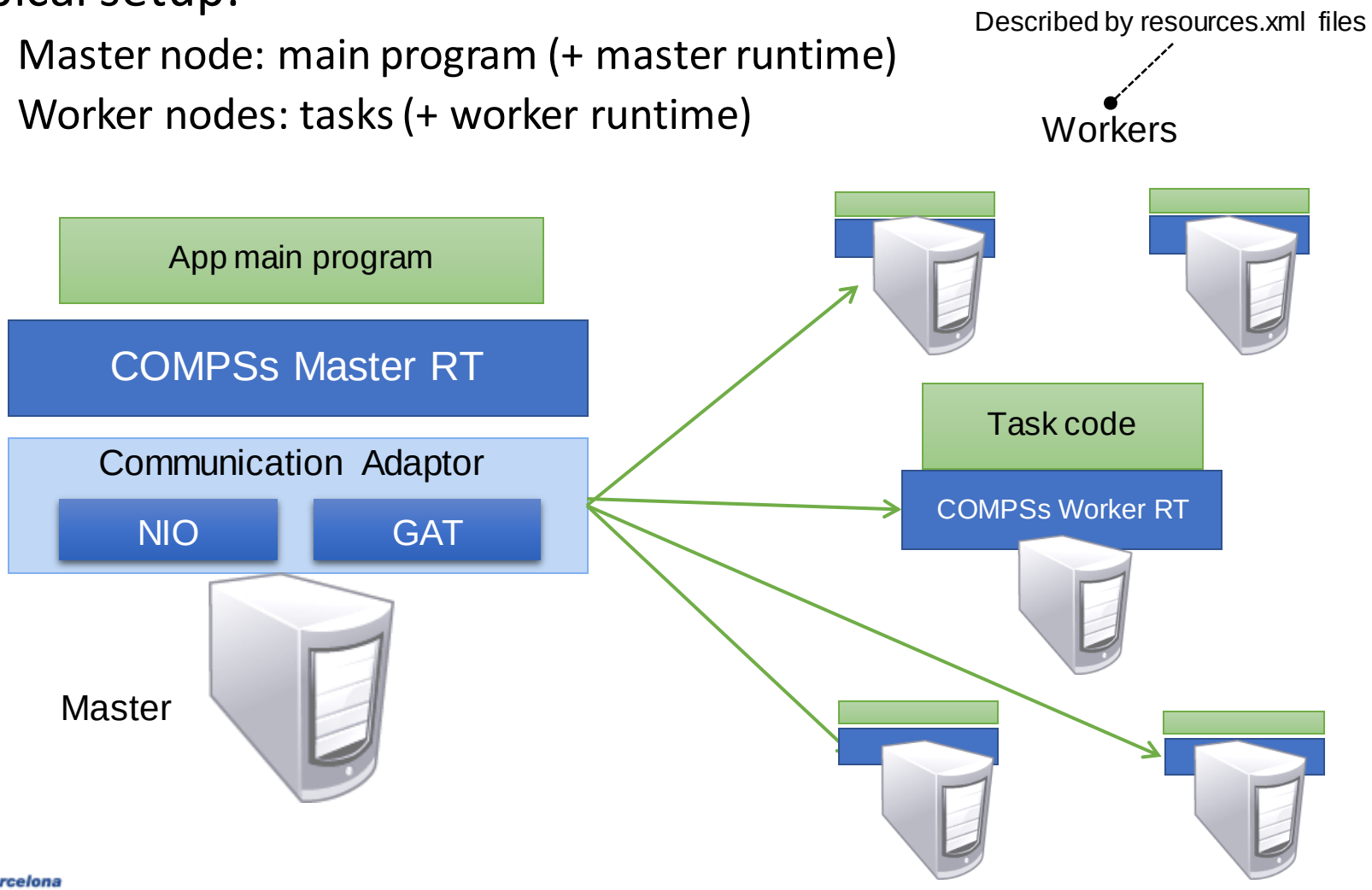
Application Exec. Desc.
- Selection of resources
- Application Code Location
- Working directory
- Provided as execution command argument

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

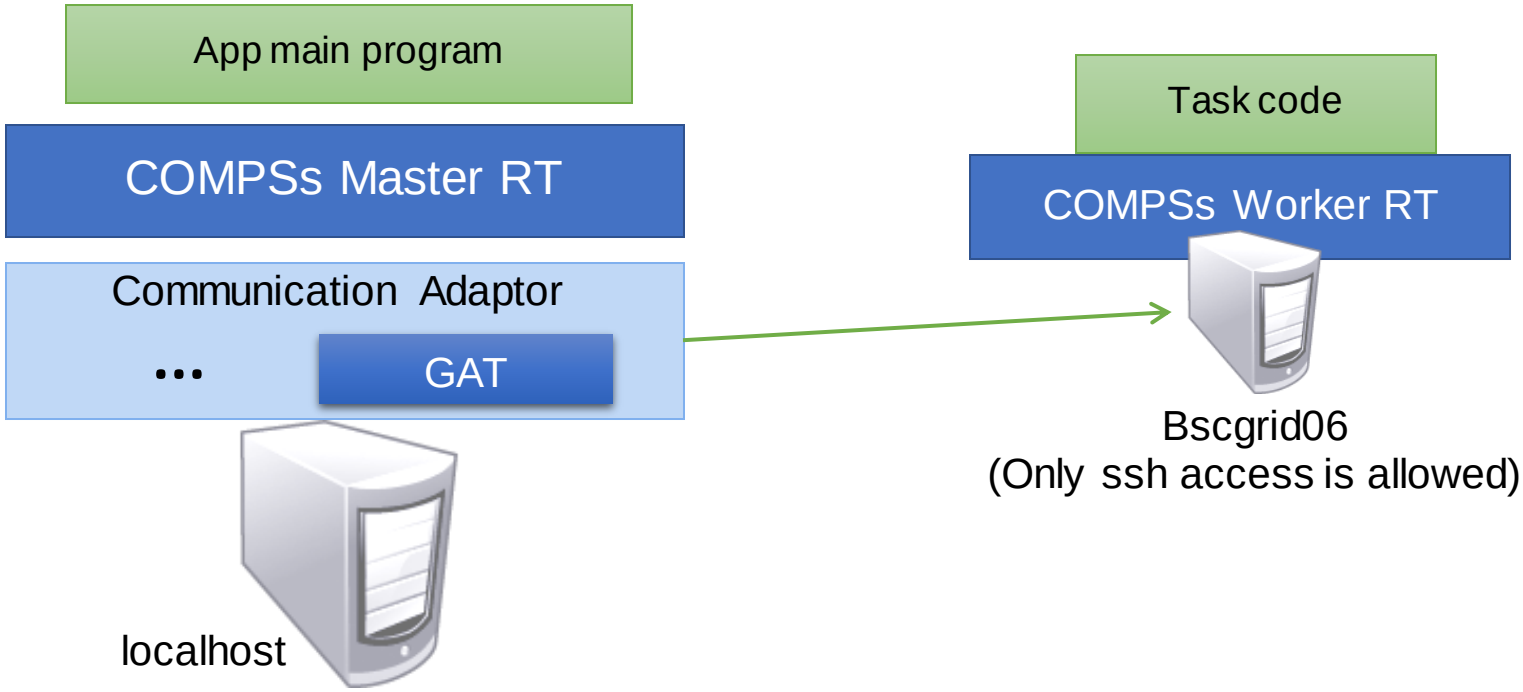
```
<?xml version="1.0" encoding="UTF-8"?>
<ResourceList>
   <!--Description for any physical node-->
   <ComputeNode Name="172.20.200.18">
     <Processor Name="P1">
        <ComputingUnits>4</ComputingUnits>
        <Architecture>amd64</Architecture>
        <Speed>3.0</Speed>
     </Processor>
     <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>
```

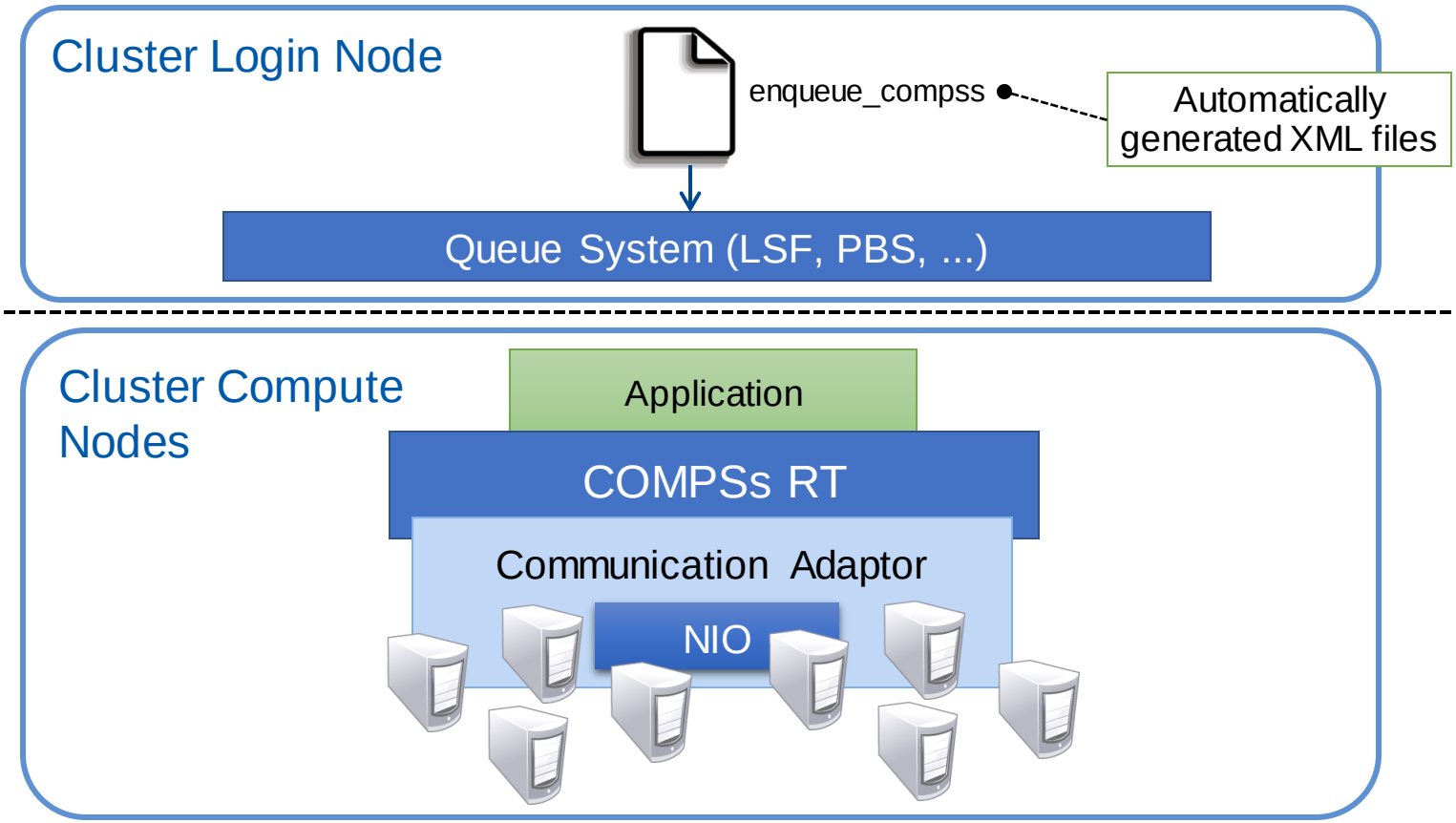
COMPSs with Interactive Hosts



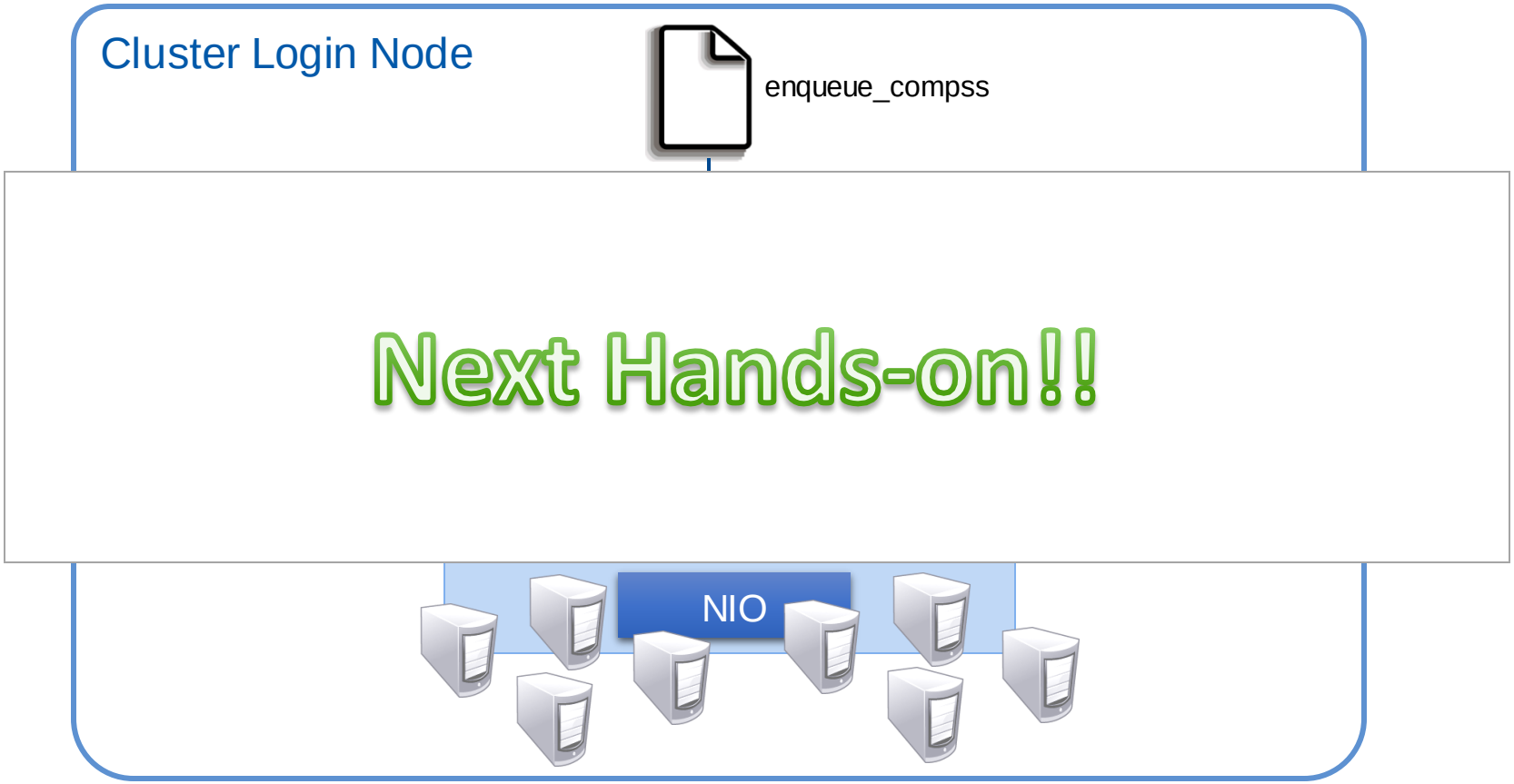
- Demo:
  - Deploy code in worker
  - Run the application with specific resources and project.xml and GAT the adaptor

COMPSs@Cluster

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

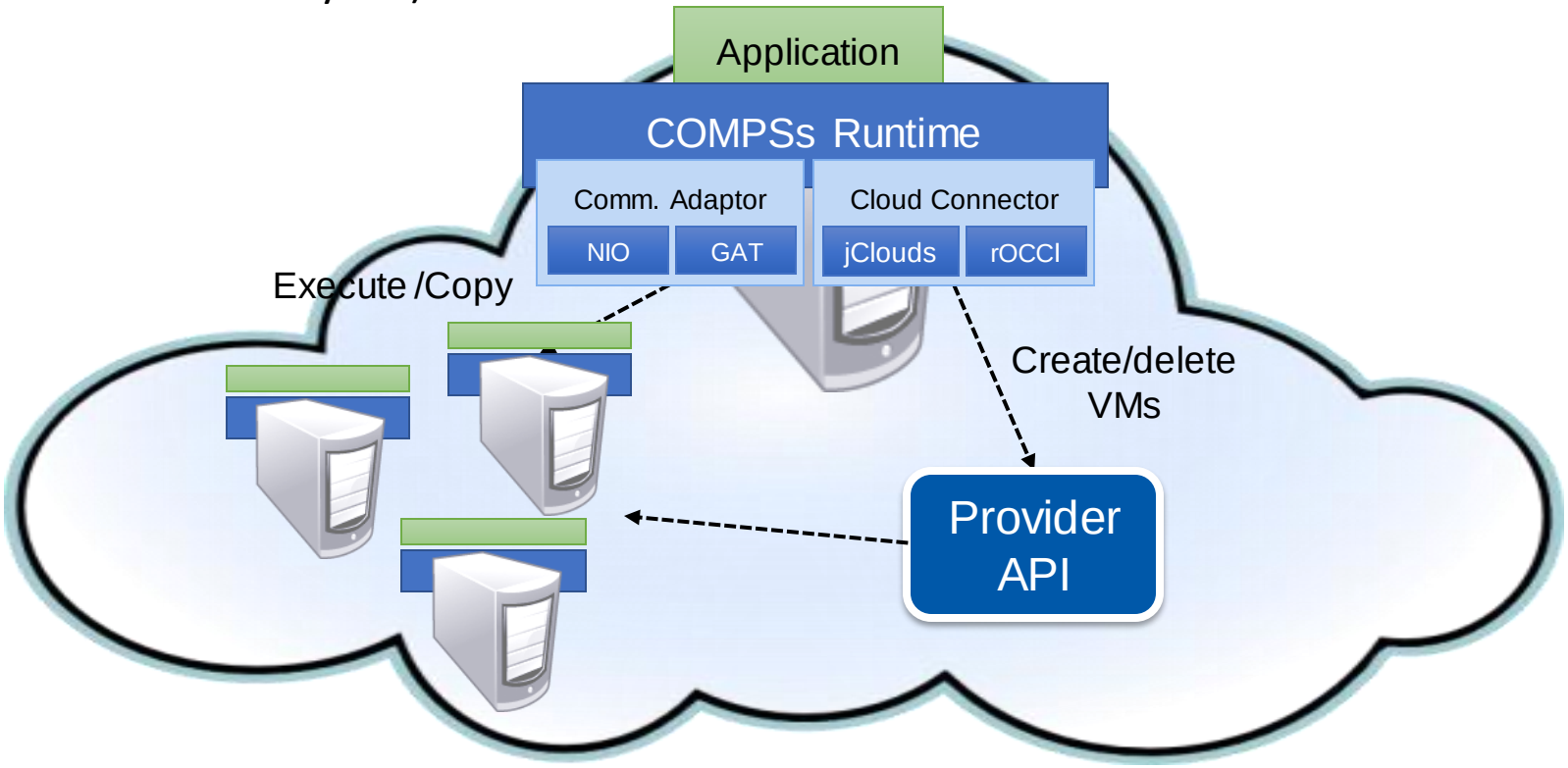


COMPSs@MN



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

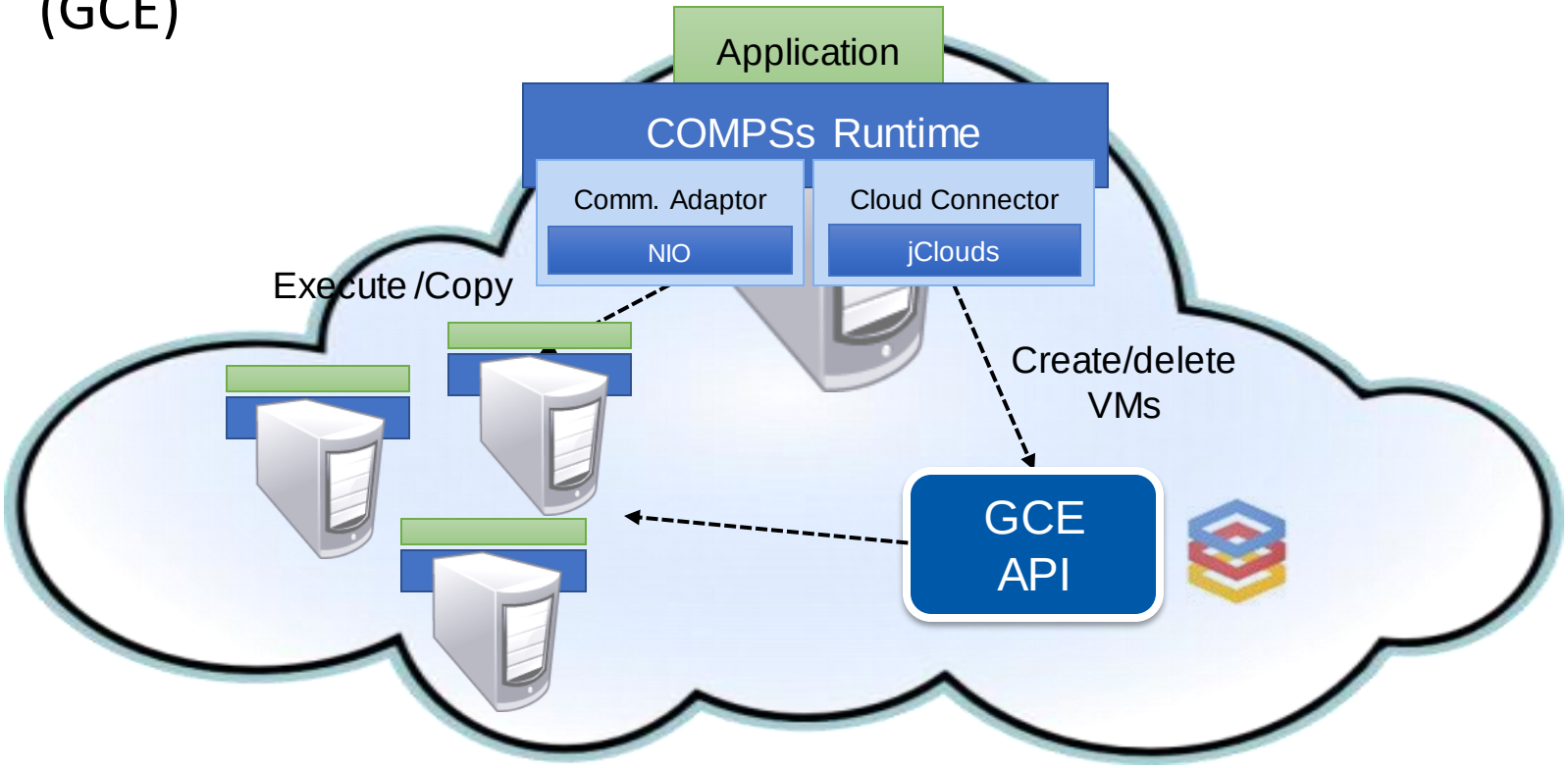
```
<ResourceList>
  <CloudProvider name="BSCCloud">
    <Endpoint>
      <Server>https://bscgrid20.bsc.es:11443</Server>
      <ConnectorJar>conn-rocci.jar</ConnectorJar>
      <ConnectorClass>es.bsc.conn.rocci.ROCCI</ConnectorClas
    </Endpoint>
    <Images>
      <Image name="debianbase">
        <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)



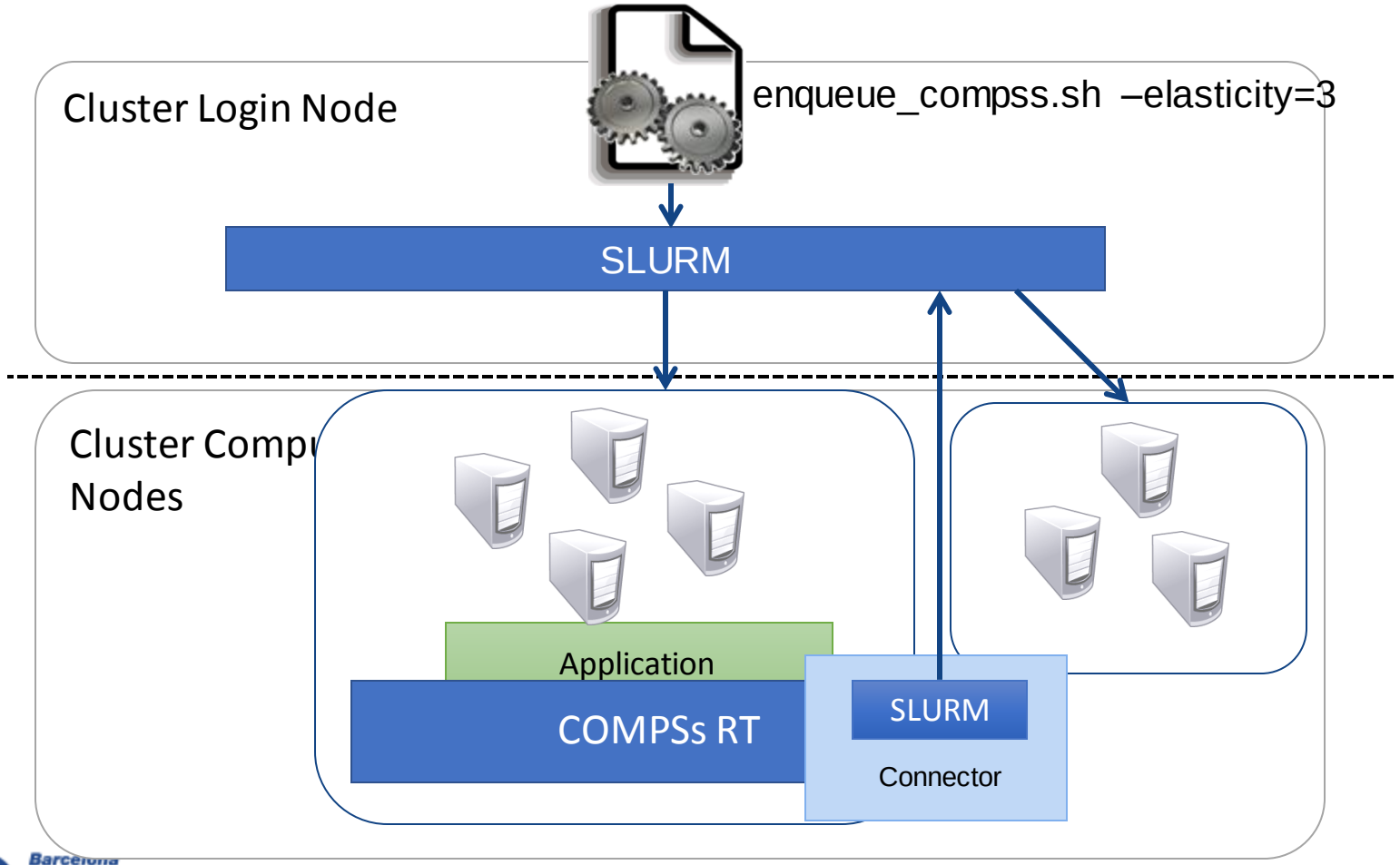
https://www.youtube.com/watch?v=XGaqUje_2zY

Advanced Execution Environment

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

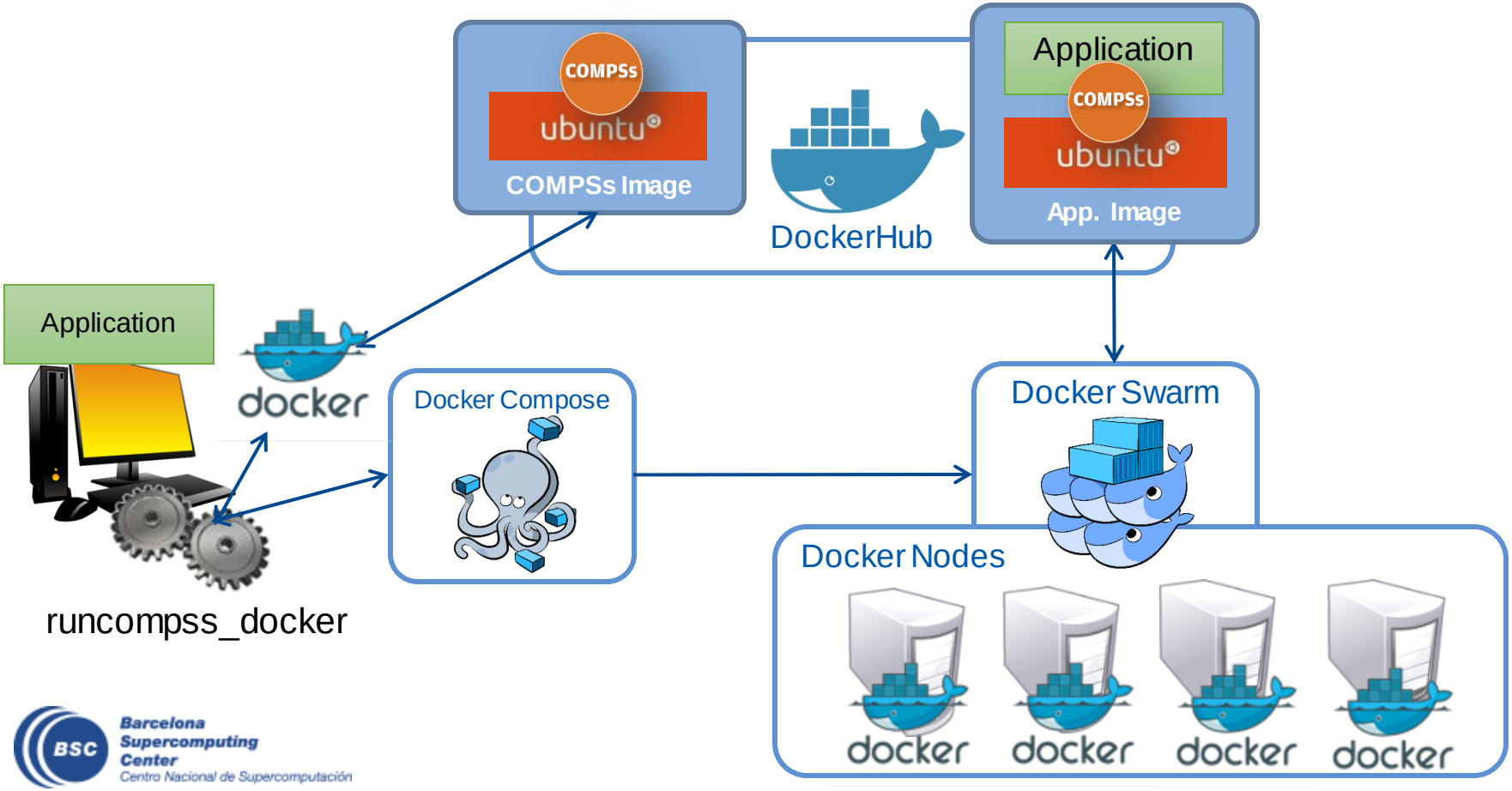
Elasticity@Clusters with SLURM Connector

- Enable the SLURM connector at submission time



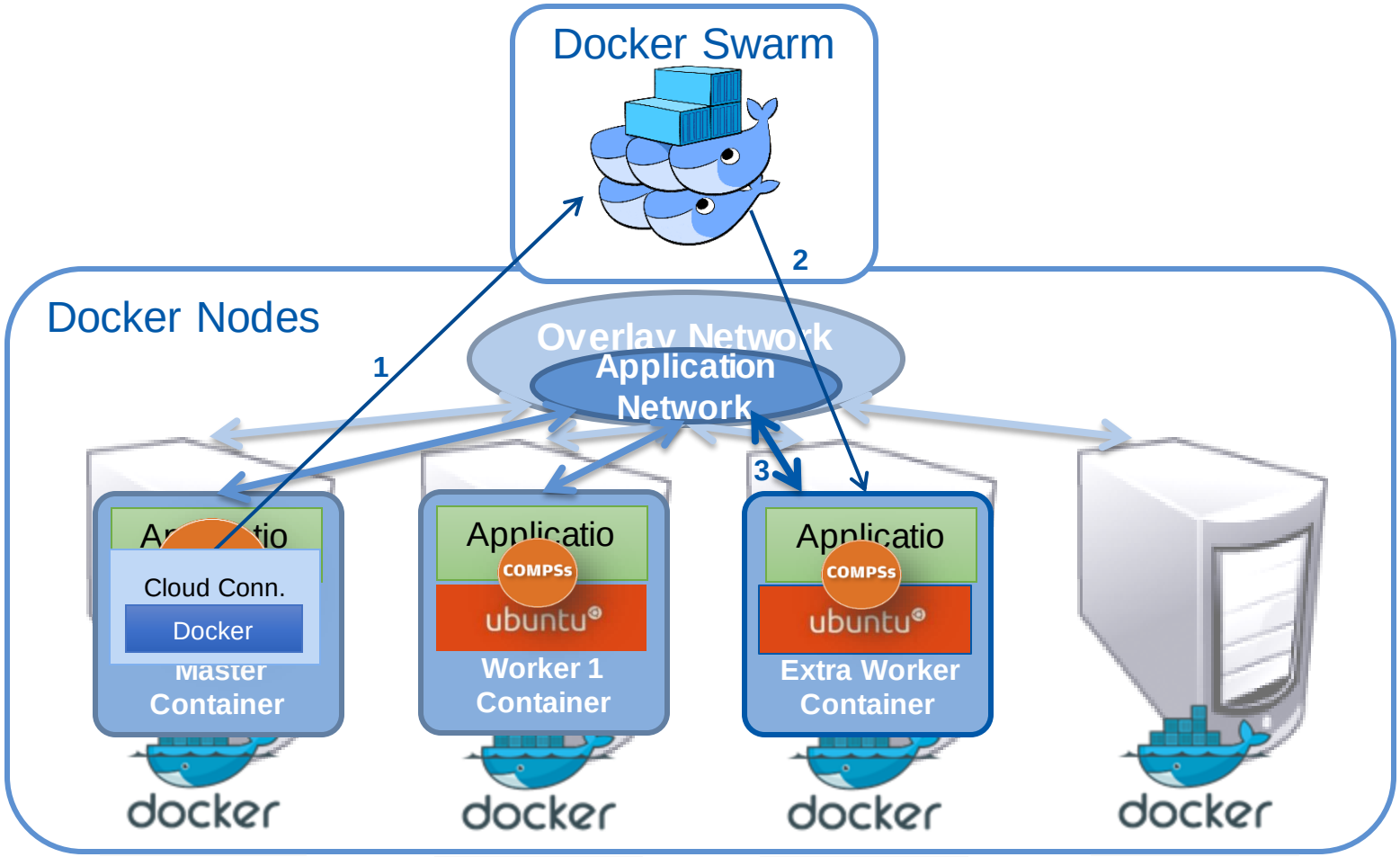
COMPSs@Docker

- 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



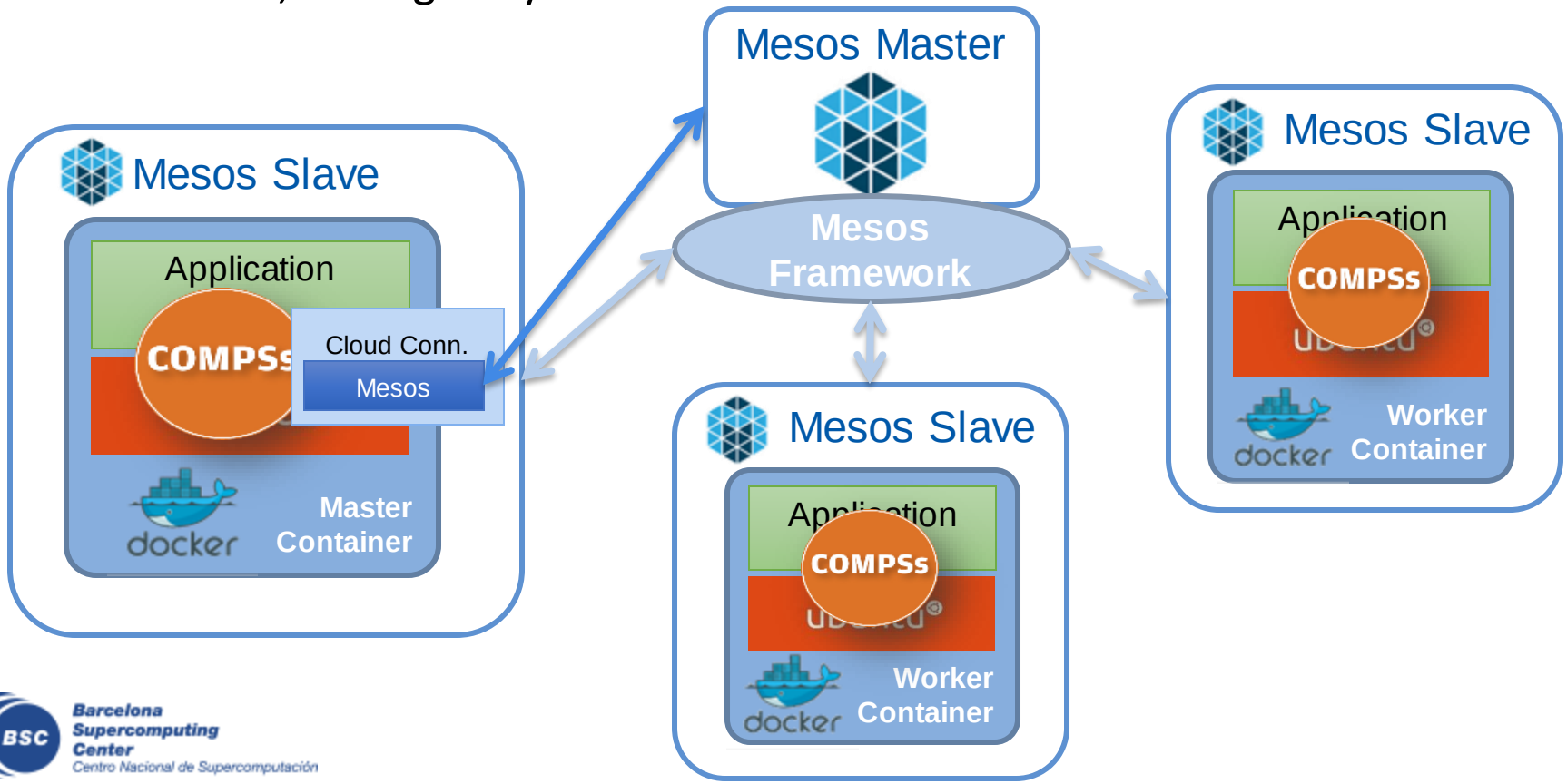
Elasticity@Docker

- Enable elasticity with the Docker connector (runcompss_docker)



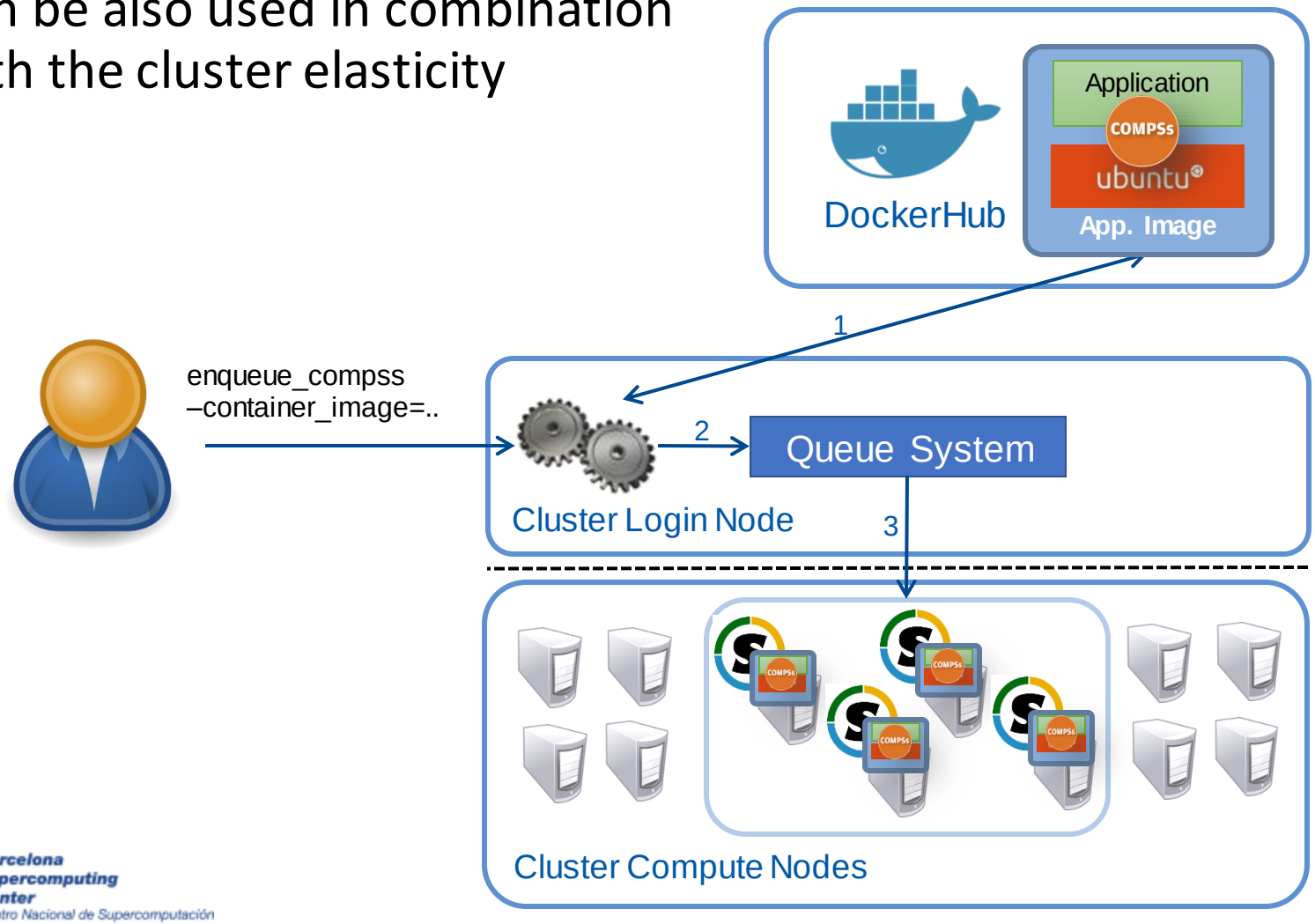
COMPSs@Mesos

- COMPSs Mesos Conector creates a Mesos Framework and negotiates the use of resources with the Mesos Master.
  - The number and type of nodes requested depends on the actual load.
  - Both the COMPSs Master and the workers are executed in Docker containers, managed by Mesos.



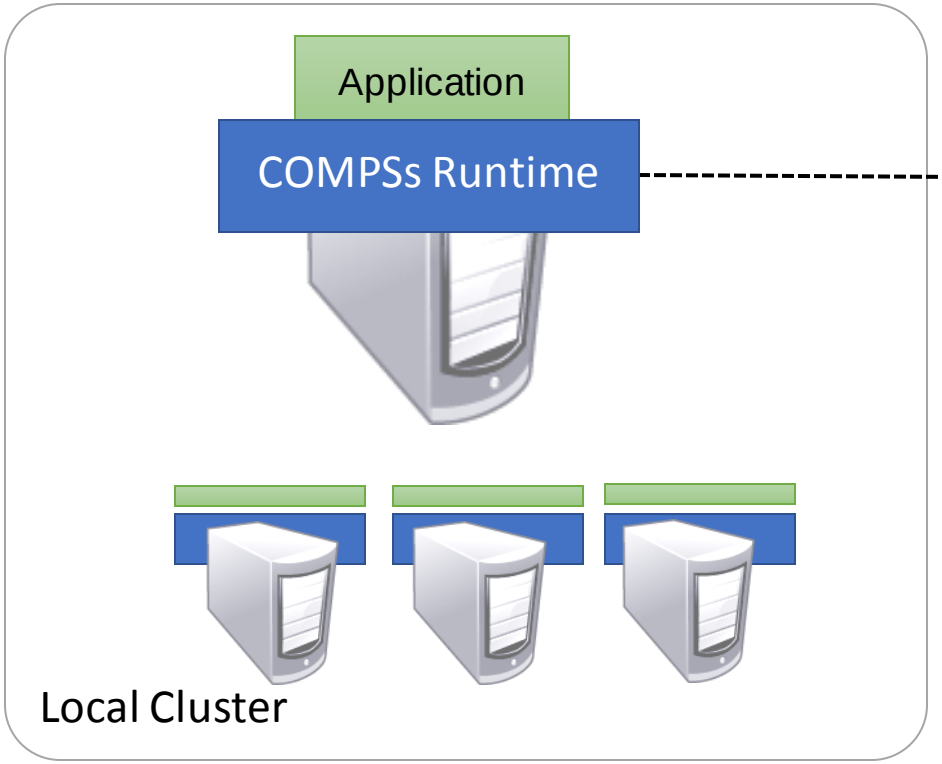
COMPSs@Singularity

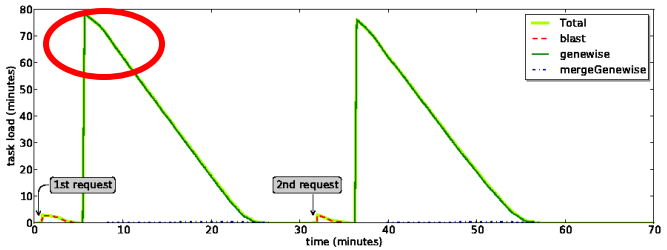
- Execute applications from a container image in HPC cluster
- Can be also used in combination with the cluster elasticity

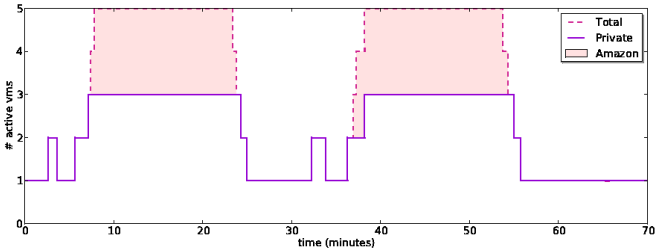


Cloud Bursting

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

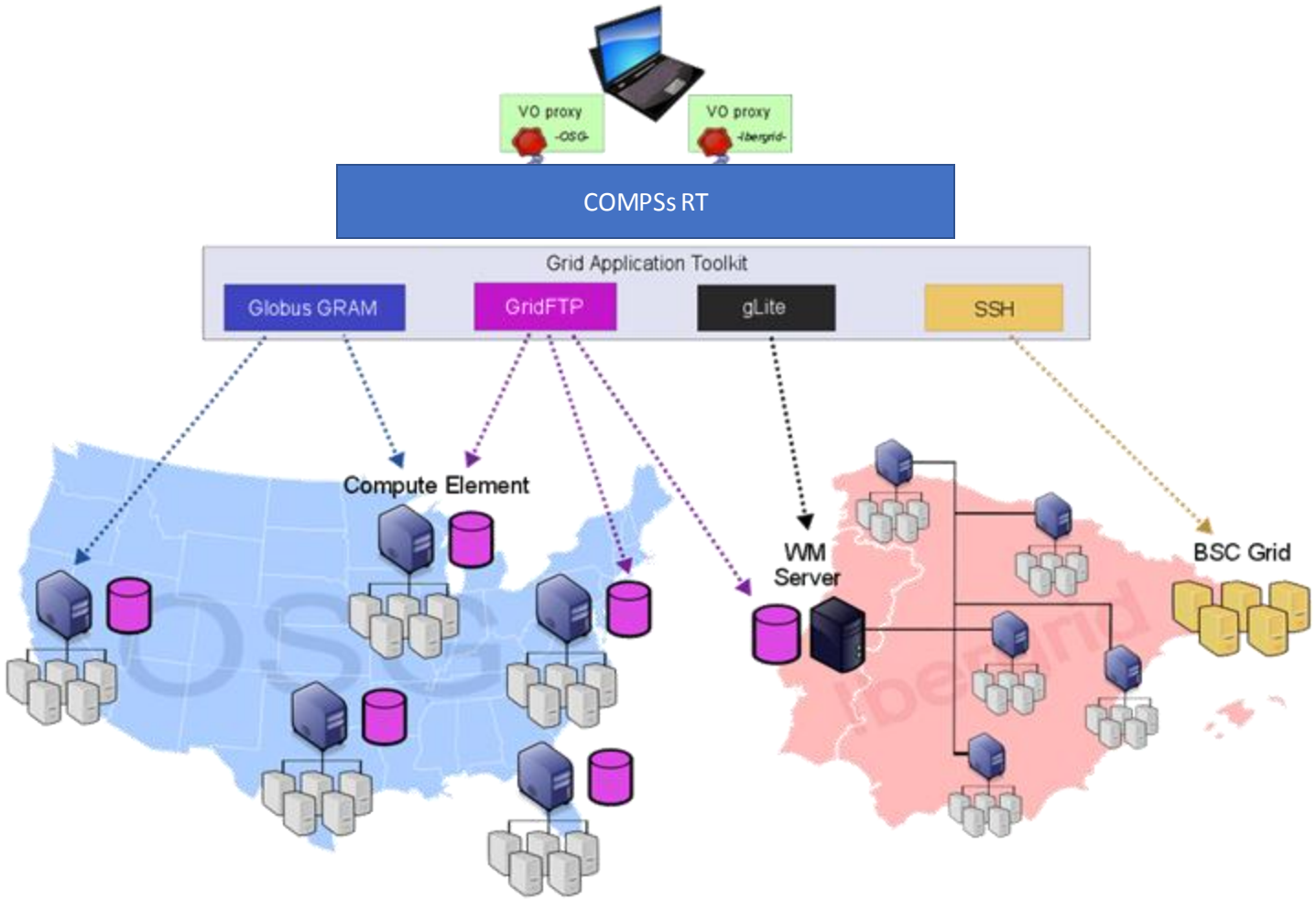






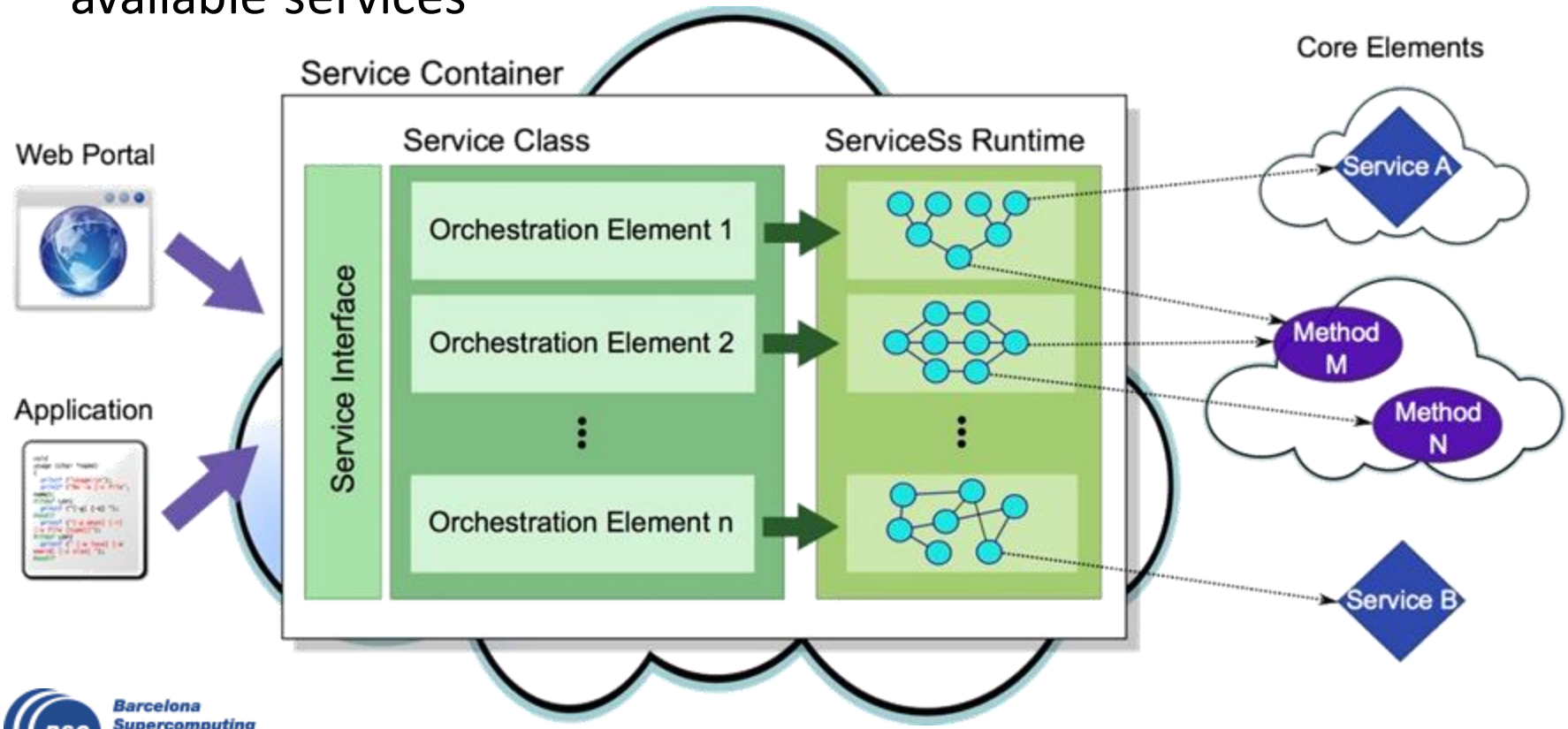
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

COMPSs@Grid


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

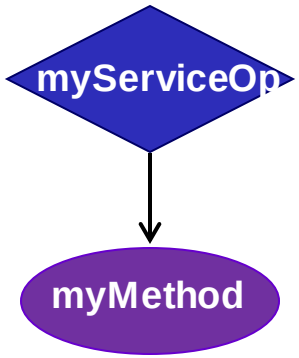
Web services with COMPSs

- A WS method implements a worklfow of tasks
- Different invocations generate different tasks
- Runtime manages the execution of the different calls in the available services



Service Operation

```
public class ServiceApp {
   @Orchestration
   public static void sampleComposite() {
      Query query = new Query(…);
      Reply reply = myServiceOp(query);

      myMethod(reply);

      reply.printToLog();
   }
}
```



Supercomputers Hands-on

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Supercomputers Hands-on

- Execution in MareNostrum
- Tracing Analysis Overview

Execution in MareNostrum

- How to connect to MareNostrum?
  - > ssh nct010XX@mn1.bsc.es (Where XX is 27 – 70)
  - Password: COMPSsAlum.0XX
- Update .bashrc
  - Edit: **.bashrc**
  - Add: "**module load COMPSs/2.2**" at the end
  - Execute: **source .bashrc**
- Where is the source code?
  - **cd**
  - **cp –r /gpfs/home/nct00/nct01026/source .**
- Where is the dataset?
  - **cp -r /gpfs/home/nct00/nct01026/dataset .**
- Available editors
  - vi
  - emacs

README

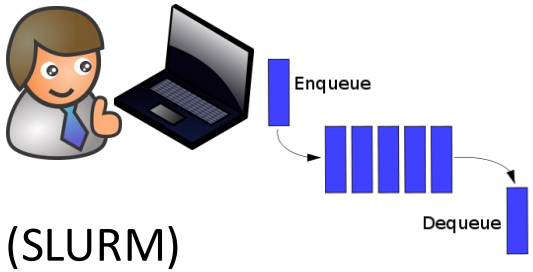
wordcount/launch_merge.sh
wordcount/launch_reduce.sh
wordcount/src/wc_merge.py
wordcount/src/wc_reduce.py
wordcount/src/wc_seq.py

kmeans/launch_kmeans.sh
kmeans/src/kmeans.py
kmeans/src/kmeans_seq.sh

WordCount@ Sequential

- Remember the dataset path
- How to launch with python sequentially?
  - > cd source/src
  - > python wordcount.py /gpfs/home/nct01/nct010XX/dataset/dataset_4f_4mb
  - Results:

```
user@login:~> python wordcount.py /path/to/dataset/
Elapsed Time (s)
0.959941864014
Words: 2551735
```



- Submit jobs to MareNostrum
  - All jobs should be submitted to the queuing system (SLURM)
  - We will use a launcher script: launch.sh
  - Useful commands:
    - squeue – This command shows the status of the job.
    - scancel jobId – This command kills a job with id 'jobId'.

Execution in MareNostrum - HandsOn

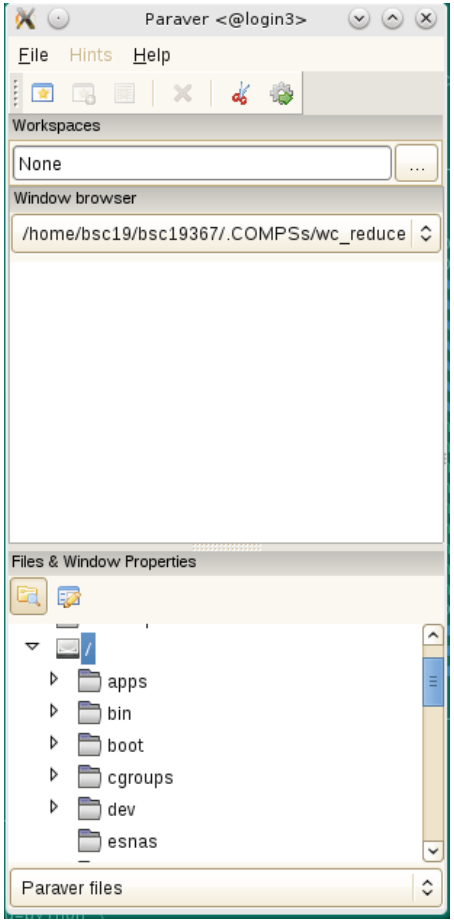
- launch_pycompss.sh

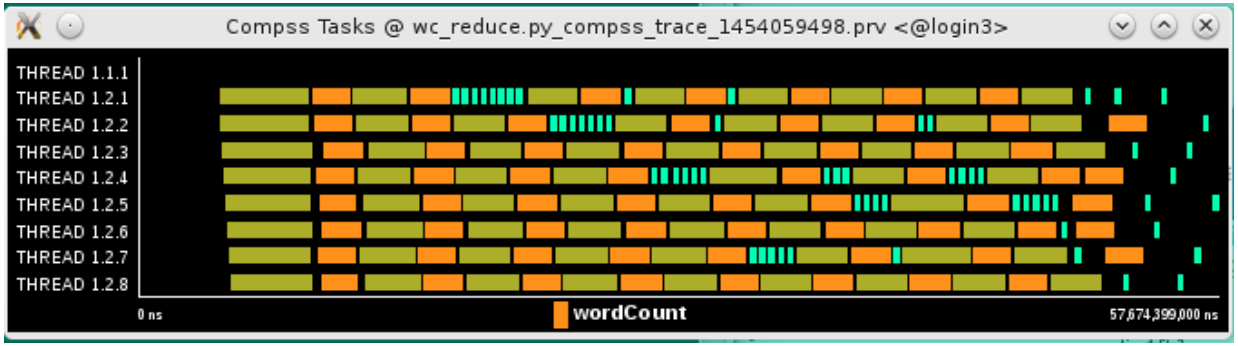
```
#/bin/bash

enqueue_compss \
  --exec_time=10 \
  --reservation=COMPSS18 \
  --num_nodes=2 \
  --lang=python \
  --tracing=true \
  --graph=true \
  /home/nct01/nct010XX/source/src/wordcount.py   /gpfs/home/nct01/nct010XX/dataset/dataset_72f_16mb

OPTION: --cpus_per_node=16 \
```

- Parameters:
  - num_nodes: amount of nodes where to execute (1 master + 1 worker).
  - cpus_per_node: amount of tasks that can be processed in parallel (1-16).
  - Dataset path: **/gpfs/home/nct01/nct010XX/dataset/dataset_72f_16mb**
- How to execute with PyCOMPSs?
  - chmod 755 launch_pycompss.sh
  - ./launch_pycompss.sh

Wordcount @ 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 trace visualization.
  - An experimented user could obtain many different views of the trace events.
- For more information about Paraver visit:
  - https://tools.bsc.es/paraver





Wordcount @ Performance Analysis

- COMPSs can generate post-execution traces of the distributed execution of the application
  - Useful for performance analysis and diagnosis
- How it works?
  - Task execution and file transfers are application events
  - An XML file is created at workers to keep track of these events
  - At the end of the execution all the XML files are merged to get the final trace file
  - COMPSs uses Extrae tool to dynamically instrument the application
    - In a worker:
      - Extrae keeps track of the events in an intermediate file
    - In the master:
      - Extrae merges the intermediate files to get the final trace file

Wordcount @ Performance Analysis

```
---------- Executing wc_reduce.py ----------
 Welcome to Extrae 3.4.3
Extrae: Generating intermediate files for Paraver traces.
Extrae: Intermediate files will be stored in /.statelite/tmpfs/gpfs/home/bsc19/bsc19000/Apps/WC/src/tutorial
Extrae: Tracing buffer can hold 500000 events
Extrae: Tracing mode is set to: Detail.
Extrae: Successfully initiated with 1 tasks

[ API] - Deploying COMPSs Runtime v2.2.rc1710 (build 20171116-1514.r3466)
[ API] - Starting COMPSs Runtime v2.2.rc1710 (build 20171116-1514.r3466)
...
[ API] - No more tasks for app 0
[ API] - Getting Result Files 0
[ API] - Execution Finished
...

Extrae: Application has ended. Tracing has been terminated.

merger: Output trace format is: Paraver
merger: Extrae 3.4.3

mpi2prv: Selected output trace format is Paraver
mpi2prv: Parsing intermediate files
mpi2prv: Generating tracefile (intermediate buffers of 745642 events)

mpi2prv: Congratulations! ./trace/wc_reduce.py_compss_trace_1453885329.prv has been generated.
------------------------------------------------------------
```

Extrae starts before the user application execution

COMPSs runtime starts

COMPSs runtime ends

The application finishes and the tracing process ends

The merge process starts

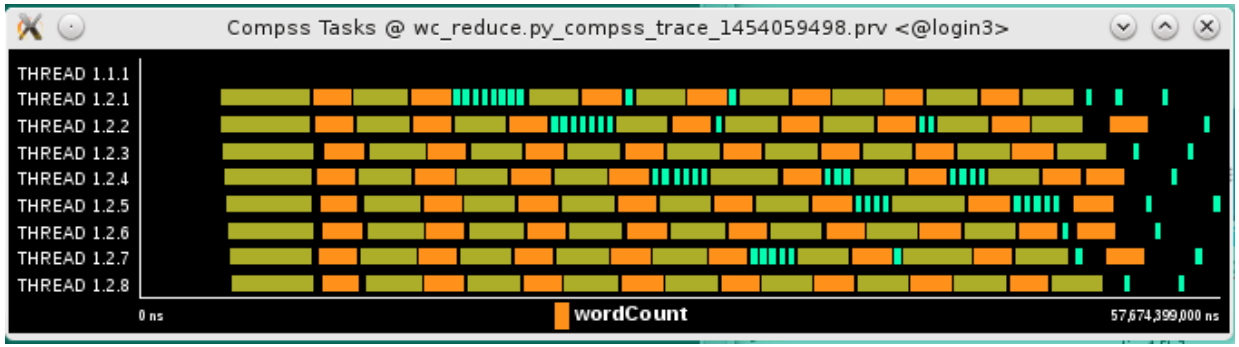
Intermediate trace files are processed

The final trace file is generated

WordCount @ Performance Analysis

- Open Paraver
  - $> module load paraver
  - $> cd $HOME/.COMPSs/wordcount.py_01
  - $> wxparaver trace/*.prv
- COMPSs provides some configuration files to automatically obtain the view of the trace
  - File/Load Configuration...

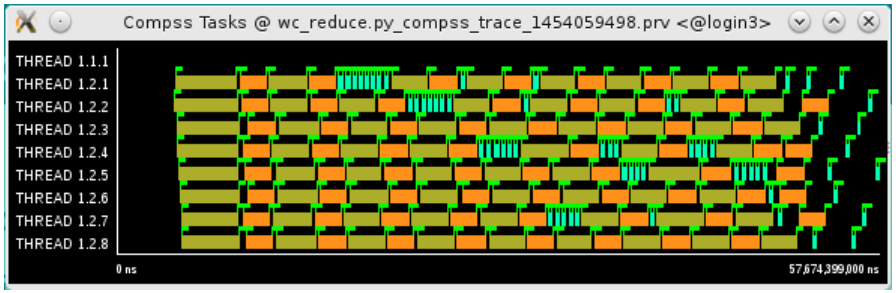
(/gpfs/apps/MN4/COMPSs/2.2/Dependencies/paraver/cfgs/compss_tasks.cfg



Wordcount @ Performance Analysis

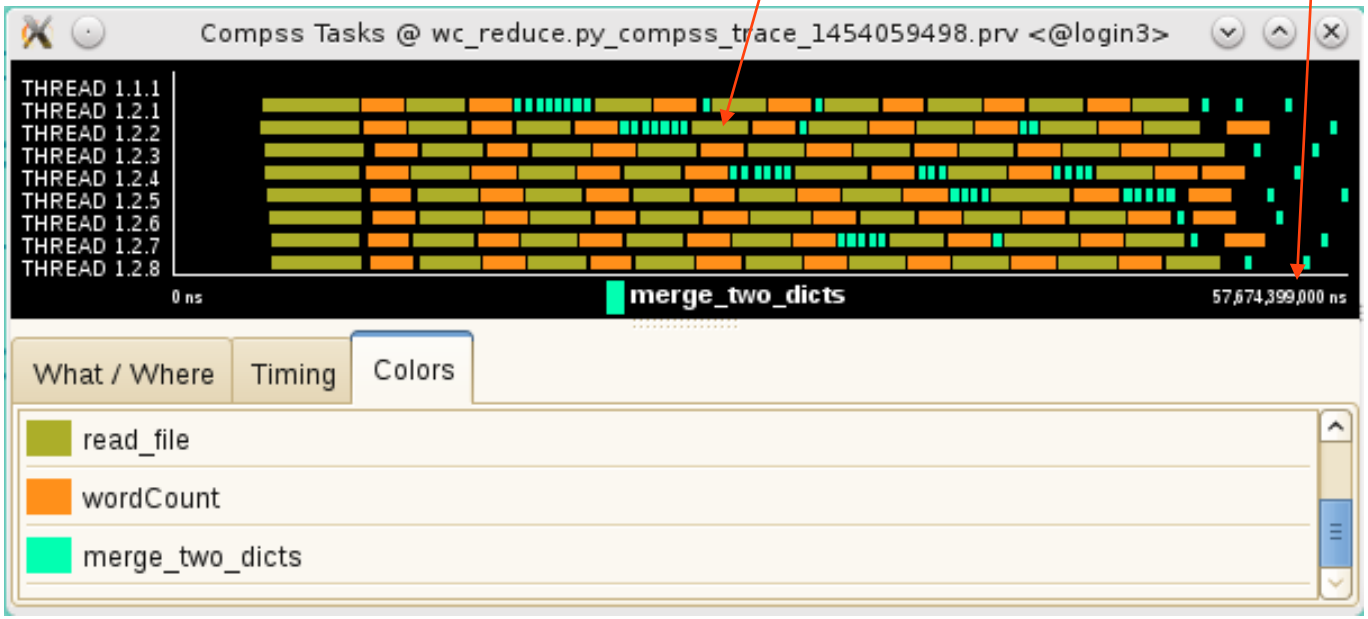
- Fit window
  - Right click on the trace window
  - Fit Semantic Scale/ Fit Both

- View Event flags
  - Right click on the trace window
  - View / Event Flags



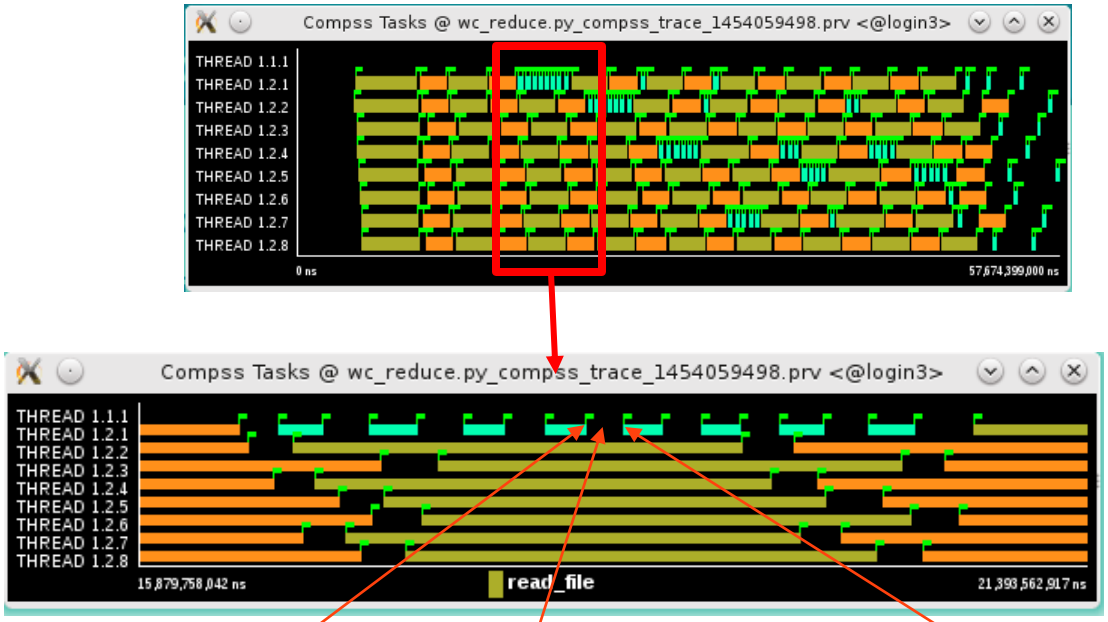
Wordcount @ Performance Analysis

- Show info Panel
  - Right click on the trace window
  - Check info panel option
  - Select Colors tab of the panel



Wordcount @ Performance Analysis

- Zoom to see details
  - Select a region in the trace window to see in detail
  - And repeat the process until the needed zoom level
  - The undo zoom option is in the right click panel



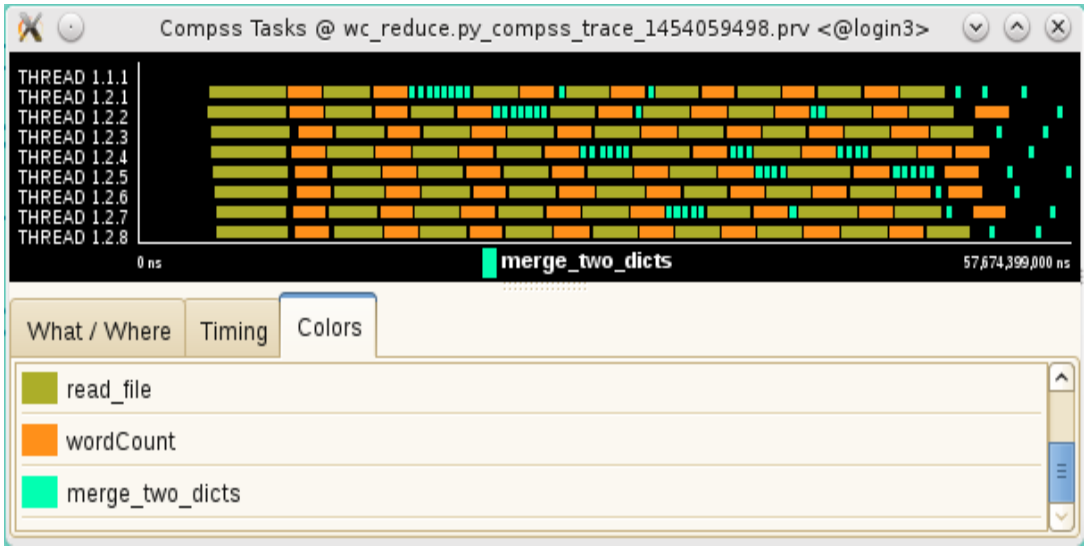
Previous task ends

Processor is idle

New task starts

Wordcount @ Performance Analysis

- Summarizing:
  - Lines in the trace:
    - THREAD 1.1.X are the master threads
    - THREAD 1.X.Y are the worker threads
- Meaning of the colours:
  - Black: idle
  - Other colors: task running
    - see the color legend
- Flags (events):
  - Start / end of task



Final notes

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Take-away messages

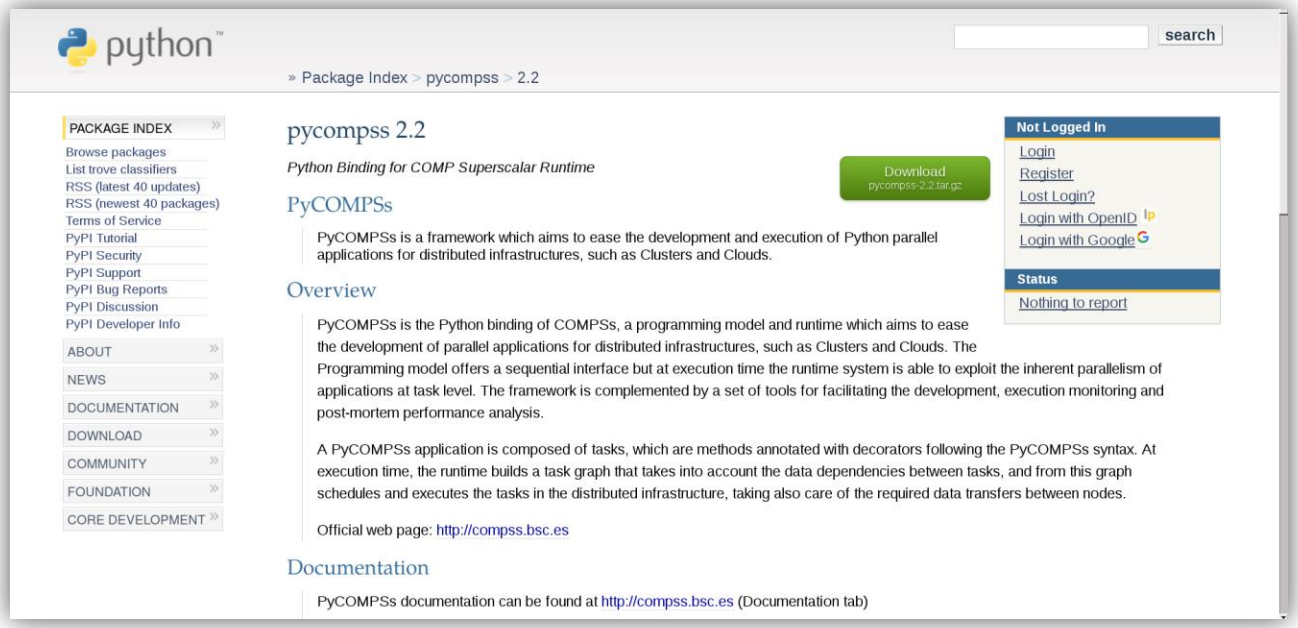
- Sequential programming approach
- Parallelization at task level
- Transparent data management and remote execution
- Can operate on different infrastructures:
  - Cluster
  - Grid
  - Cloud (Public/Private)
    - PaaS
    - IaaS
  - Containers
  - Web services

Further Information

- Project page:
  - http://www.bsc.es/compss
- Direct downloads page:
  - https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/downloads
    - Virtual Appliance for testing & sample applications
    - Tutorials
    - Red-Hat & Debian based installation packages
    - Source Code
- Application Repository
  - http://compss.bsc.es/projects/bar/wiki/Applications
    - Several examples of applications developed with COMPSs

PyCOMPSs - PIP install

- Release of PyCOMPSs pip package to enable automatic installation with "pip install".
- Available since January 2017
  - https://pypi.python.org/pypi/pycompss/2.2/
- Documentation for the package



Join us


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