www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

COMPSs Tutorial

February 19th 2015, Barcelona

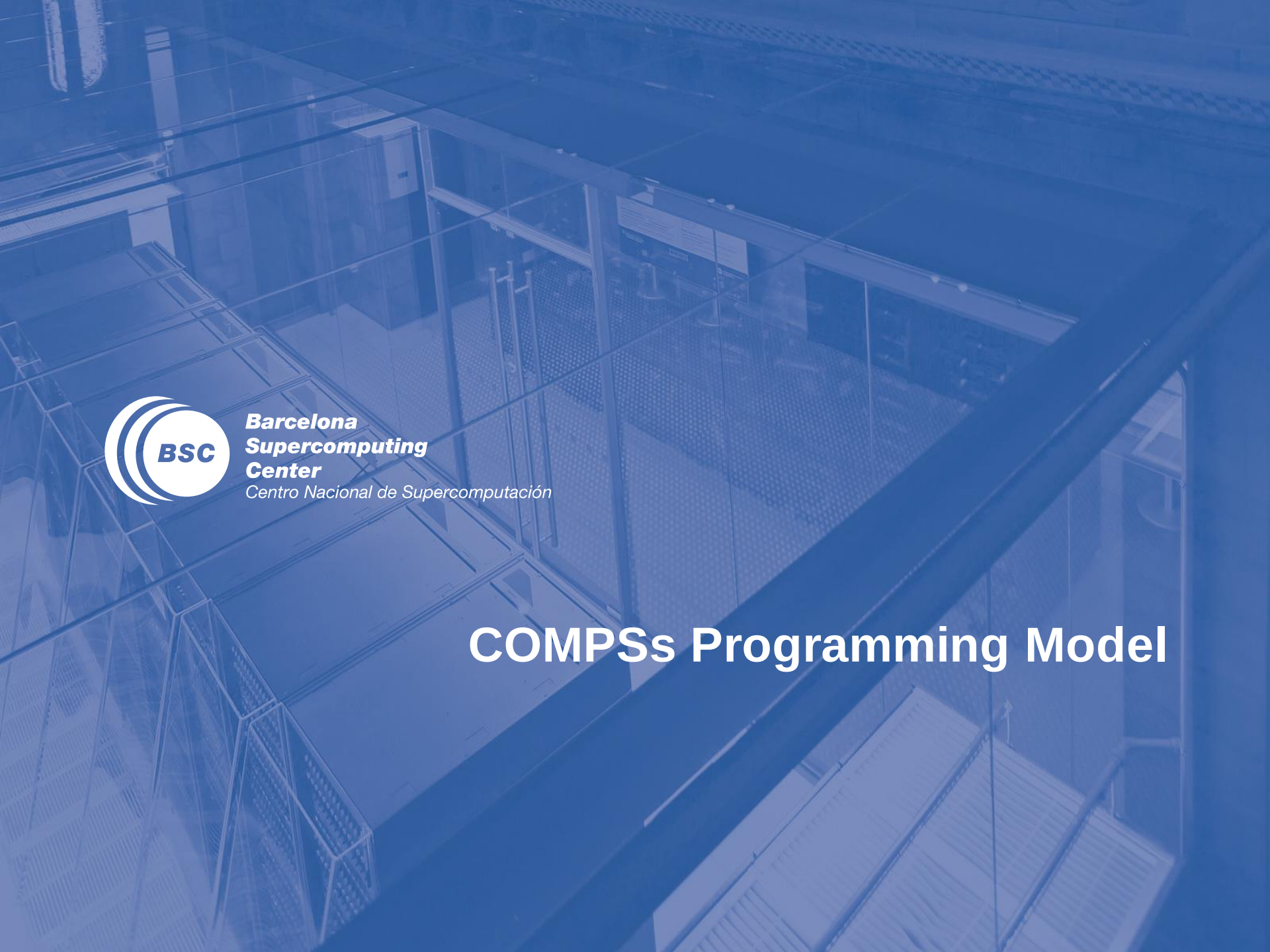
Rosa M. Badia, Carlos Diaz, Jorge Ejarque
Daniele Lezzi, Francesc Lordan
Roger Rafanell, Raül Sirvent, Cristian Ramon-Cortes,
Fredy Juarez

Outline (Feb 19th 2015)

- Roundtable(9:00 - 9:30): Presentation and background of participants
- Session 1 (9:30 – 11:00): Introduction to COMPSs
  - Programming model
    - Overview
    - Steps
    - Properties
  - COMPSs runtime system
    - Overview
    - Features
- Coffee break (11:00 – 11:30)
- Session 2 (11:30 – 13:00): Application examples
  - Sample codes & Demos
    - Matmul (Java and Python)
    - Gene Detection Service (Deployment with IDE)

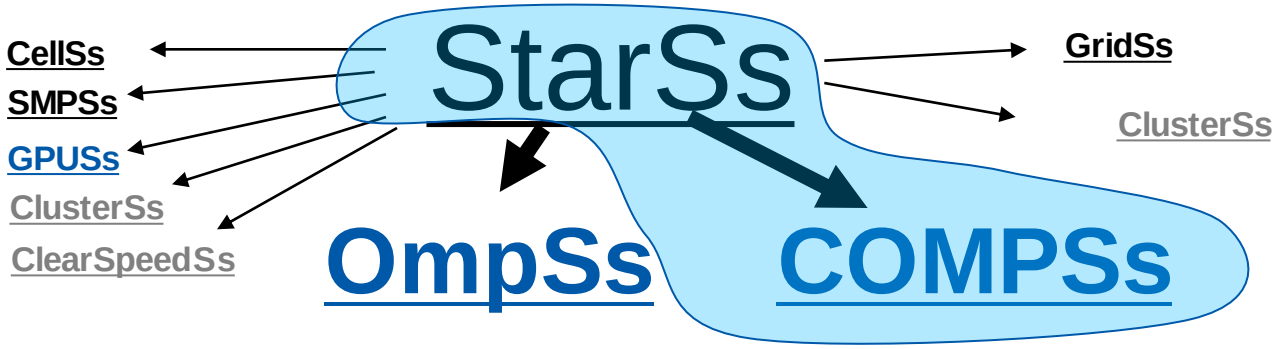
Outline (Feb 19th 2015)

- Lunch Break (13:00 -14:00)
- Session 3 (14:00 - 15:30) Hands-on I
  - Virtual Machine Setup
  - Java Hands-on
    - HMMER overview & code modification
    - Configuration, monitoring, debugging
    - Overview of tracing, trace analysis
- Coffee break: 15:30 – 16:00
- Session 4 (16:00 – 18:00): Hands-on II
  - Python Hands-on
  - IDE Hands-on
- Final notes



COMPSs Programming Model

StarSs Programming Model

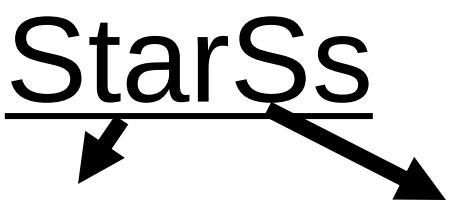


- StarSs
  - Sequential code + Annotations
  - Task-Based
  - Simple linear address space
  - Support for SMP, GPUs, Cluster

- OpenSource
  - http://compss.bsc.es

- Programmability/Portability
  - "Same" source code runs on "any" machine
  - Incremental parallelization/restructure
  - Focus in the problem, not in the hardware
- Performance
  - Intelligent Runtime
  - Locality awareness.
  - Automatic detection of parallelism
  - Matches computation to resource features

StarSs Programming Model: Granularities



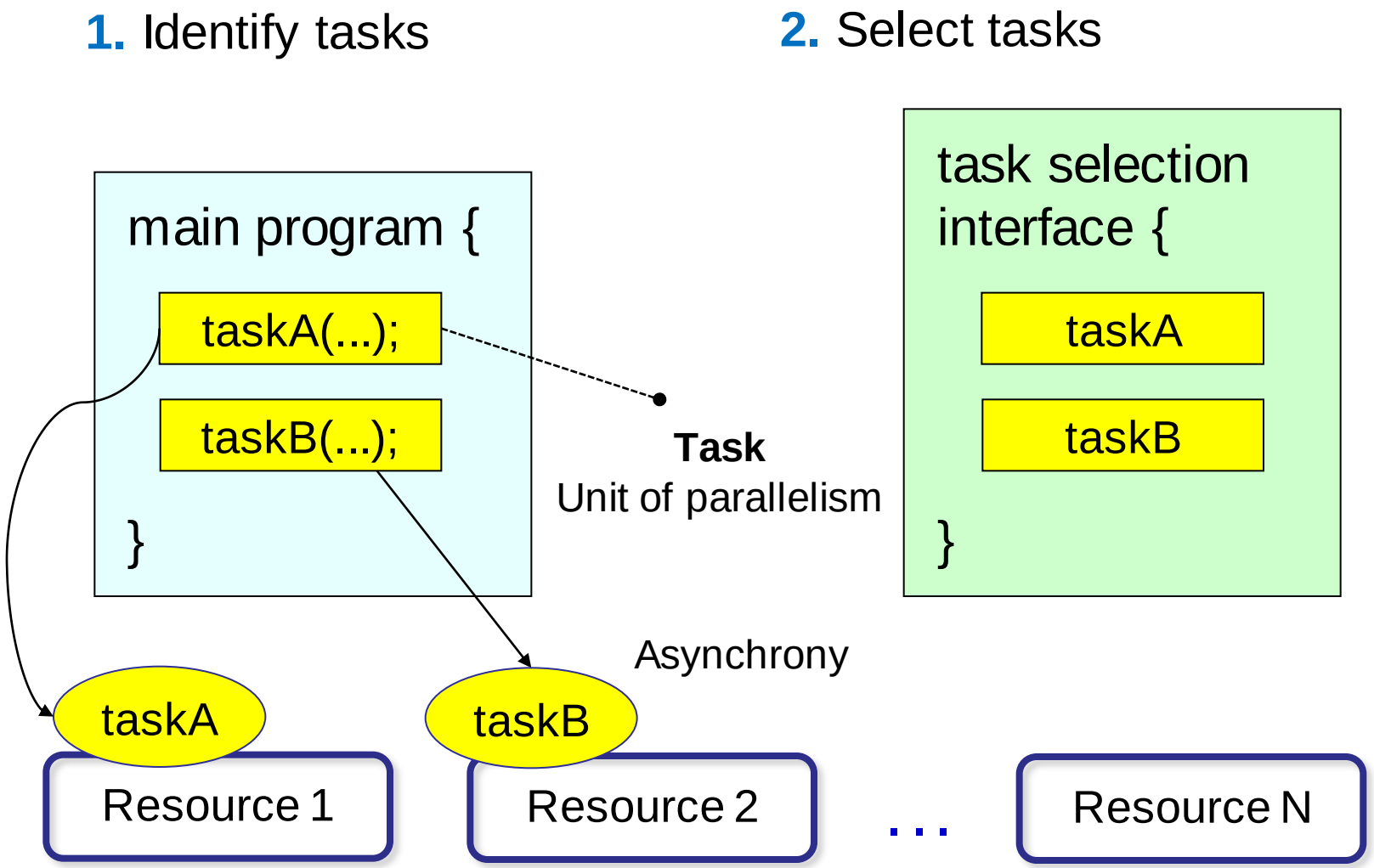
OmpSs	COMPSs
Average Task Granularity: – 100 microseconds – 10 milliseconds	1 second – 1 day
Address space to compute dependencies: – Memory	Files, Objects
Language bindings: – C, C++, FORTRAN	Java, C, Python

Overview: 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

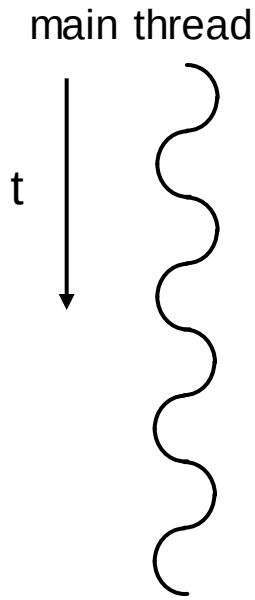
- Target applications: composed of tasks, most of them repetitive
  - Granularity of the tasks or programs
  - Data: files, objects, arrays and primitive types

Programming Model: Steps

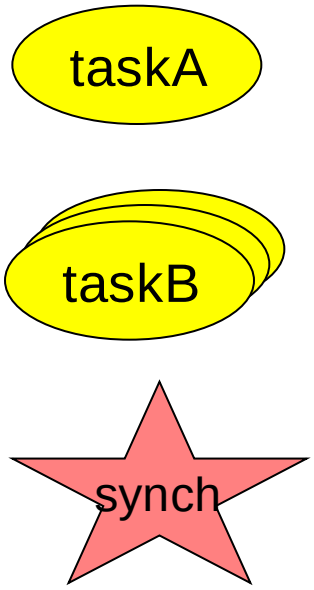


Programming Model: Properties (I)

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




```
Main Program {
   taskA(data1);

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

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

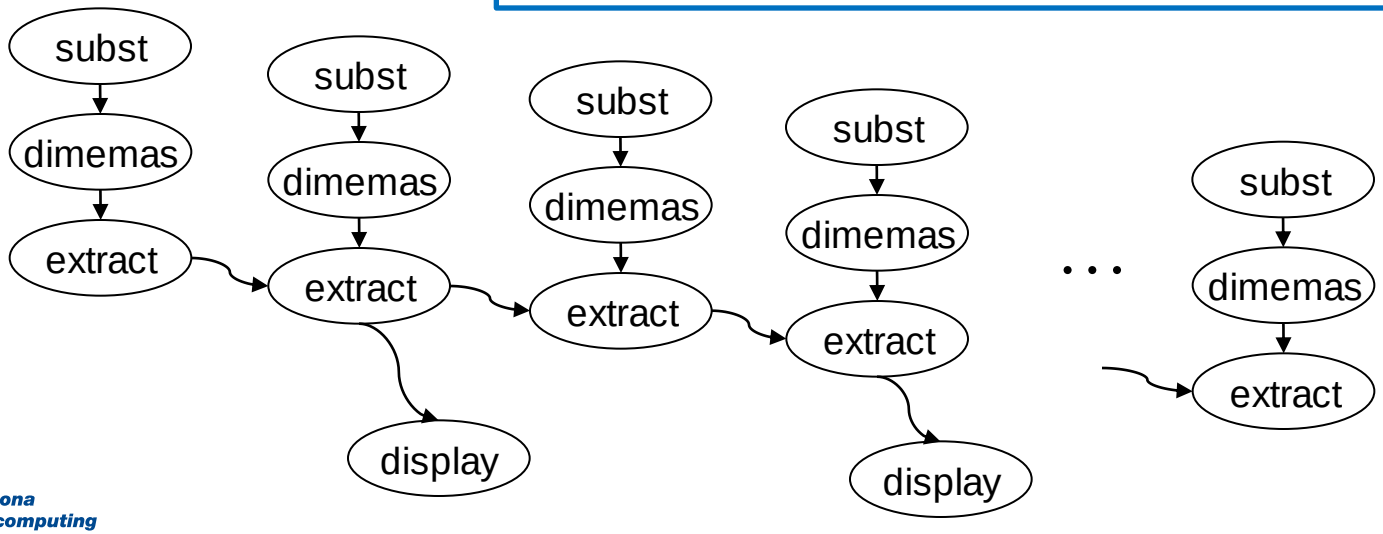


Programming Model: Dependency detection

- Automatic on-the-fly creation of a task dependency graph

Main Program

```
for (int i = 0; i < N; i++) {
   newBWD = random();
   subst(refCFG, newBWD, newCFG);
   dimemas(newCFG, trace, dimOUT);
   extract(newBWD, dimOUT, finalOUT);
   if (i % 2 == 0) display(finalOUT);
}
```



Programming Model: Properties (II)

- Infrastructure unaware

Application

Task Selection Interface

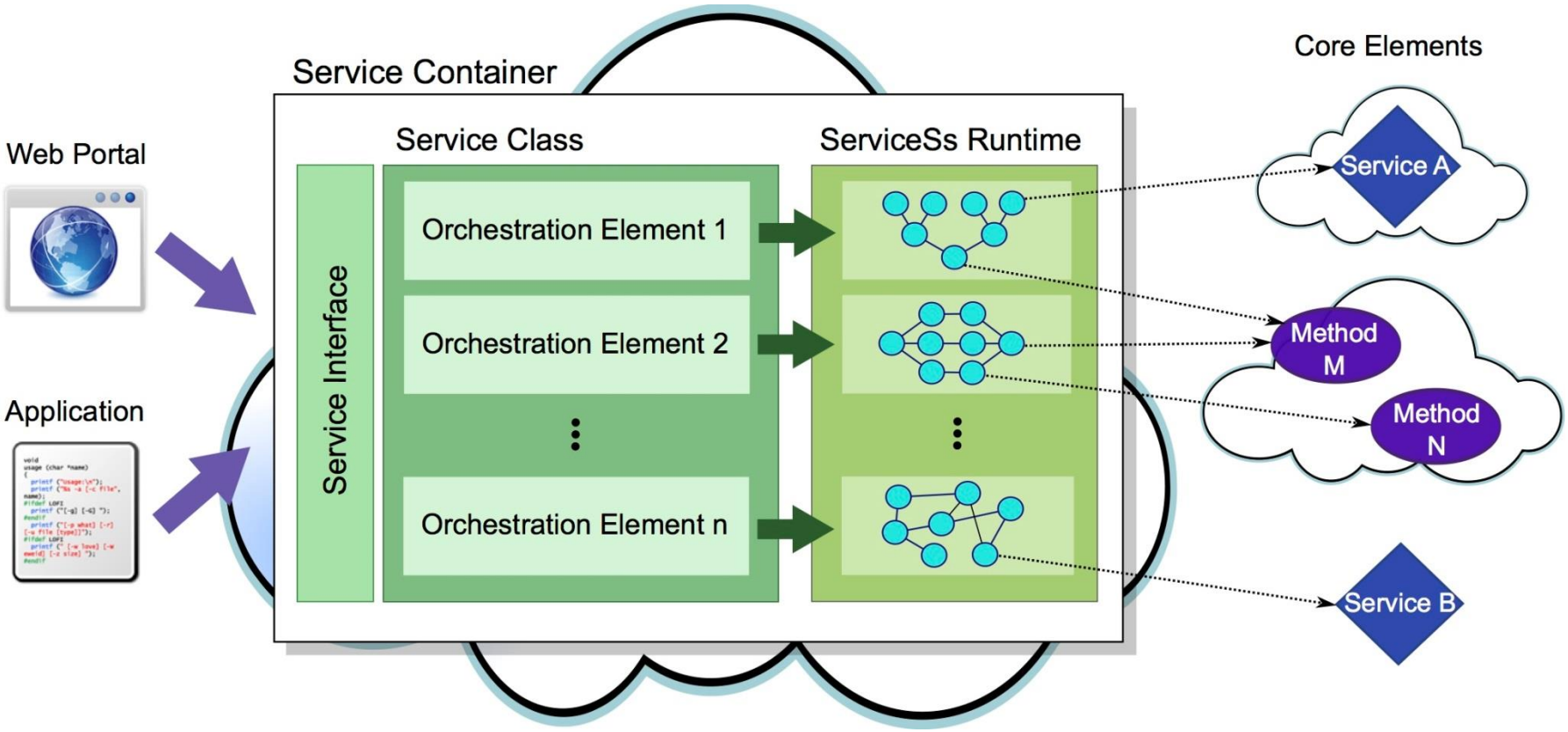
Grid

Cluster

Cloud

Programming Model: Properties (III)

- Method and Services composition



Programming Model: Task selection interface

```
public interface SampleItf {
   @Constraints(processorCPUCount = 1, memoryPhysicalSize = 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
   );
}
```

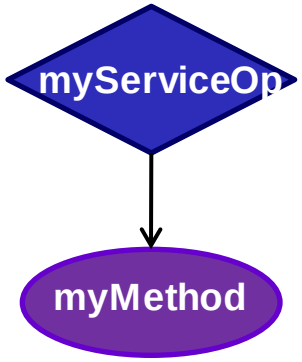
Programming Model: Regular Main program

```
public class App {

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

       myMethod(reply);

       reply.printToLog();
   }
}
```

Service task call

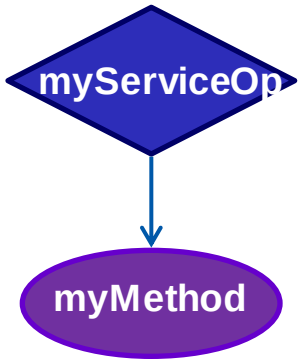
Method task call

Synchronization

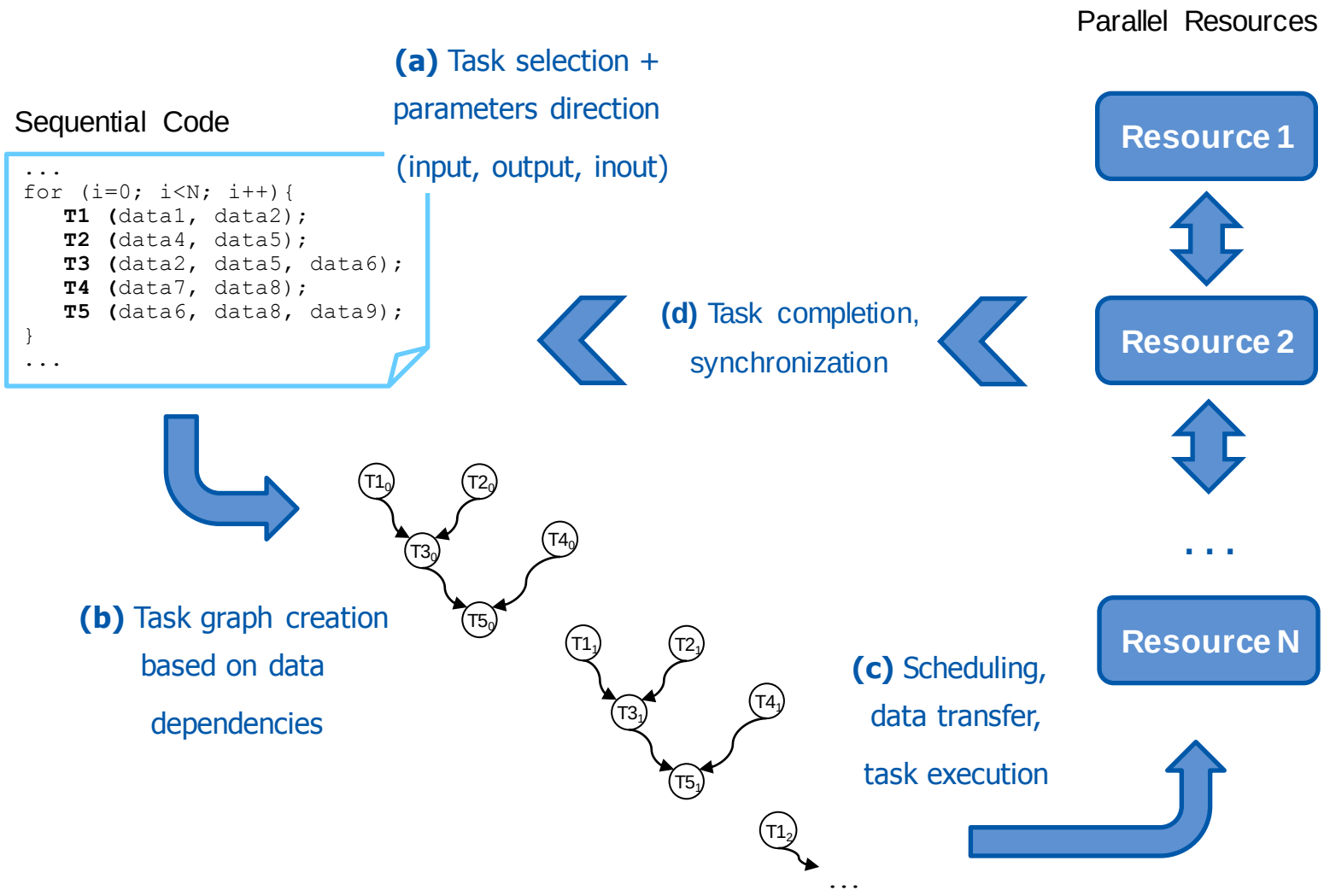


Programming Model: Service Operation

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

       myMethod(reply);

       reply.printToLog();
   }
}
```



Programming Model: Summary



Programming Model: 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);
        }
}
```

- Subroutine

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

Programming Model: Sample App (Interface)

- Task Annotation Interface

```
public interface SimpleItf {

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

}
```

Implementation

Parameter metadata

Programming Model: Sample App (Main Program)

- Main program NO CHANGES!

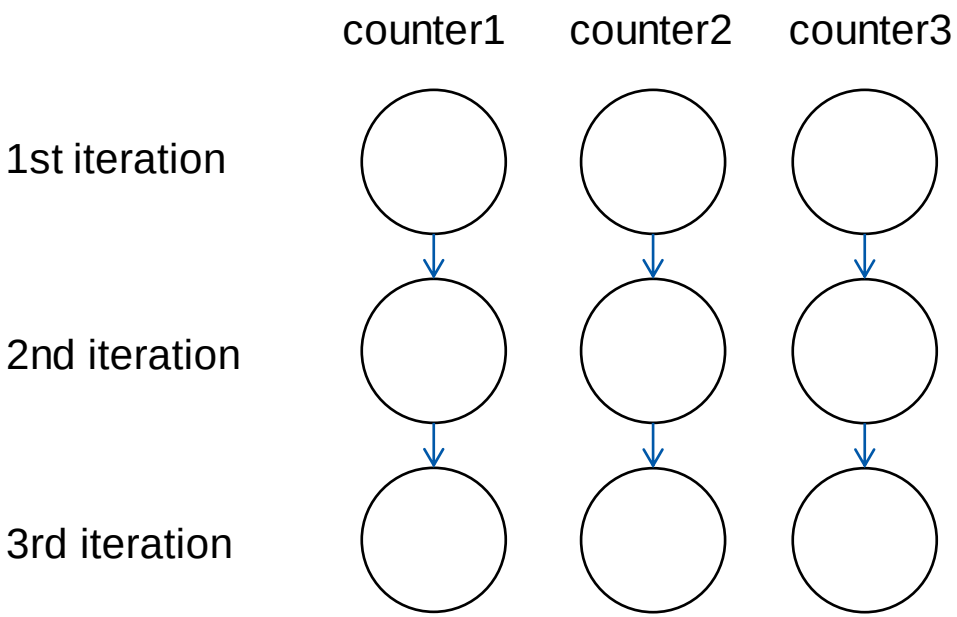
```
public static void main(String[] args) {
        String counter1 = args[0], counter2 = args[1],
               counter3 = args[2];

        initializeCounters(counter1, counter2, counter3);

        for (i = 0; i < 3; i++) {
          increment(counter1);
          increment(counter2);
          increment(counter3);
        }
}
```

Programming Model: Task Graph

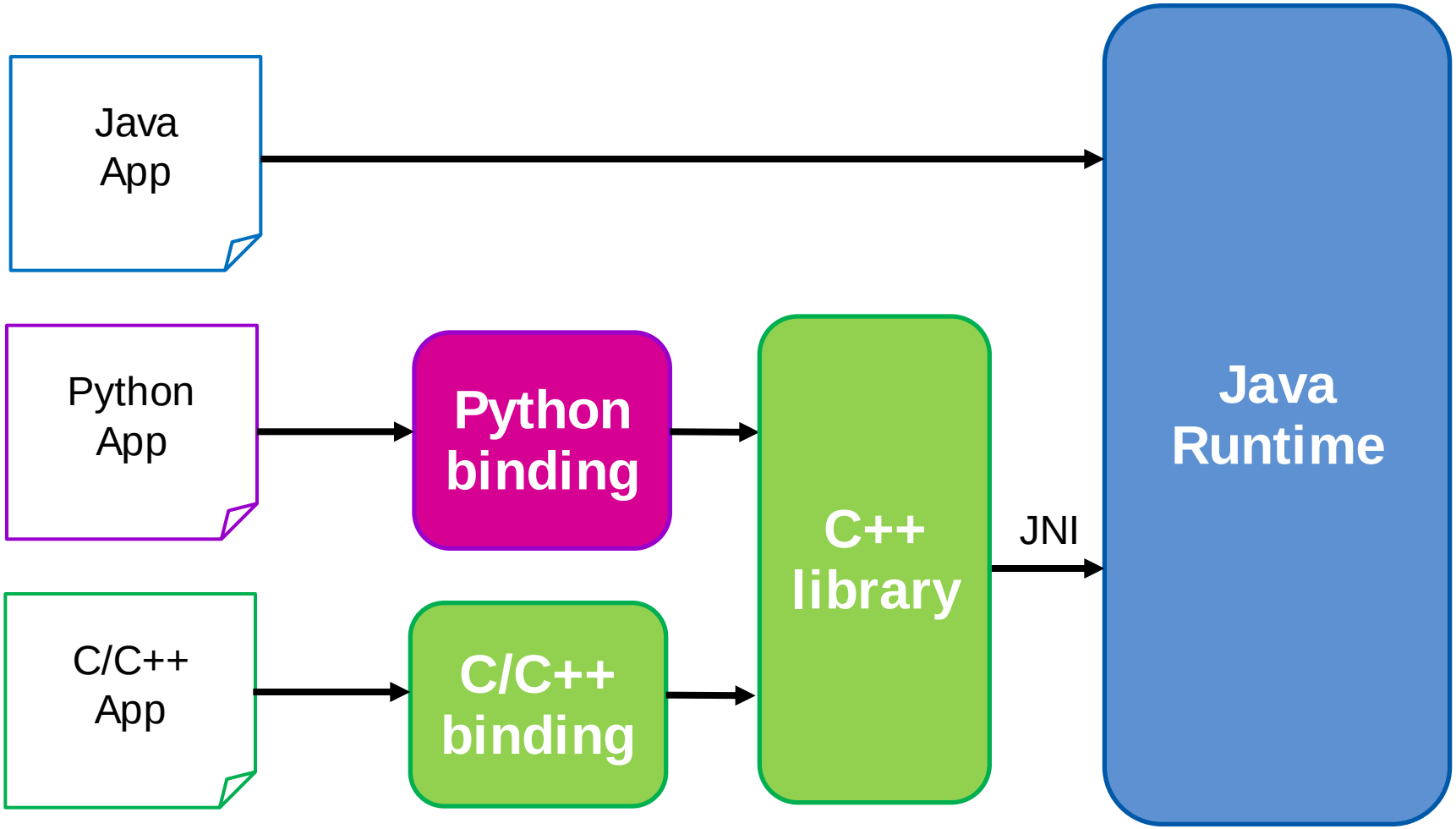
Main Loop

```
for (i = 0; i < 3; i++) {
        increment(counter1);
        increment(counter2);
        increment(counter3);
}
```

Task Graph

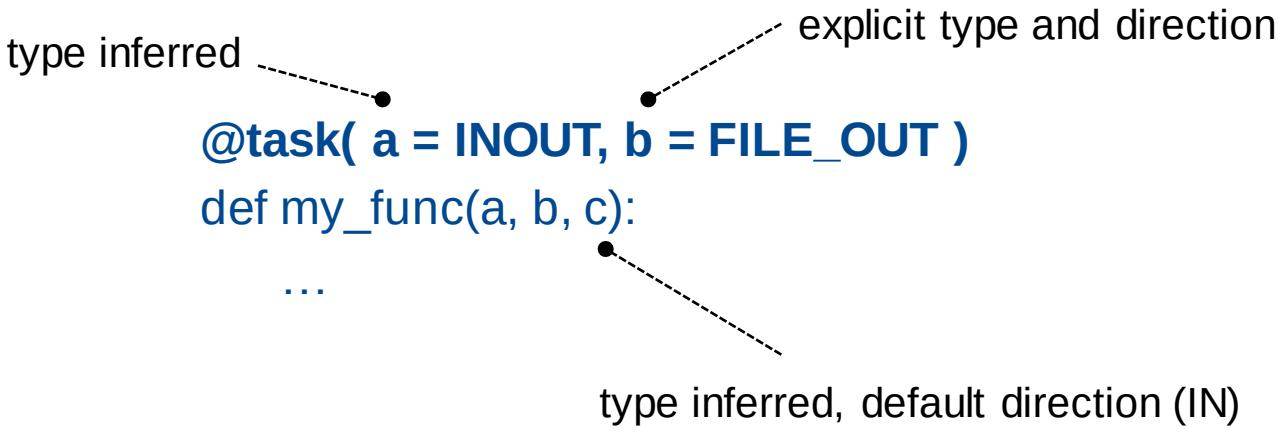


COMPSs Bindings



PyCOMPSs: Task definition

- Task definition with Python decorators
  - Provide information about task parameters (*TYPE_DIRECTION*):
    - Type
      - Only mandatory for files
      - Inferred for the rest of the types
    - Direction
      - Default IN (read-only)
      - Mandatory for INOUT (read-write) and OUT (write-only)



PyCOMPSs: Task definition (II)

- The @task decorator: special arguments
  - Type of the return value → mandatory if a value is returned

```
@task(returns = int)
def ret_func():
    return 1
```

  - Does the task modify the callee object? → default True

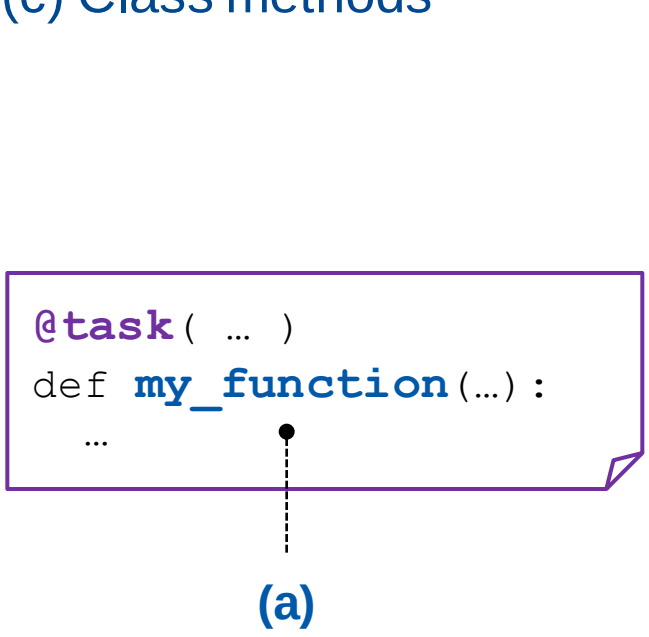
```
class MyClass(object):

    @task(isModifier = False)
    def instance_method(self):
        … # self is NOT modified here
```

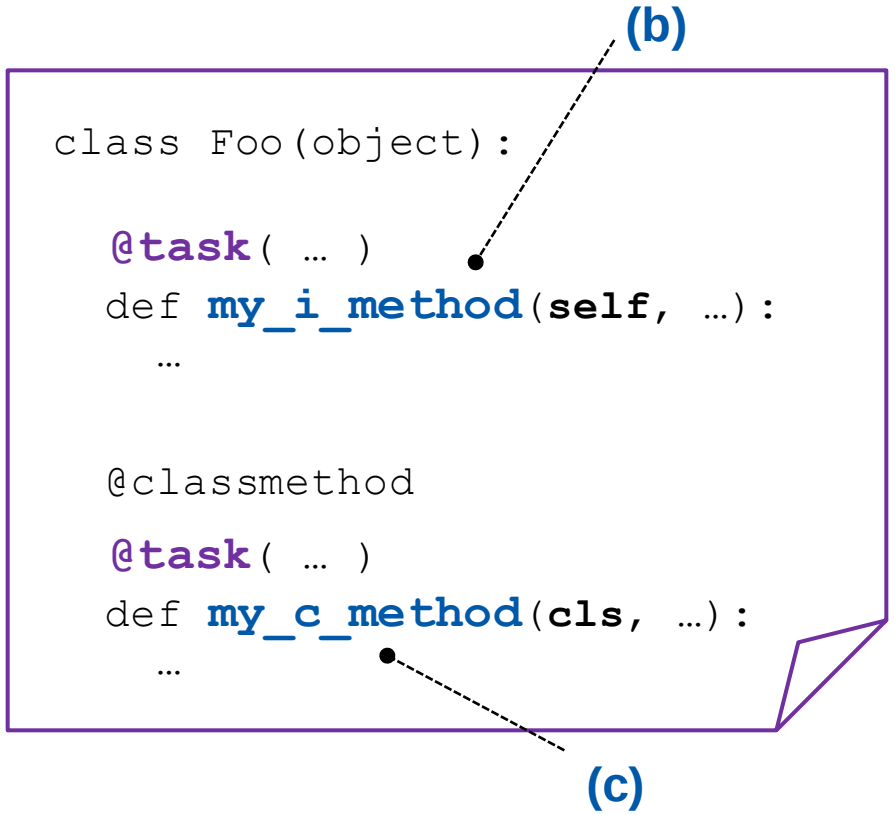
  - Is it a priority task? → Default False

```
@task(priority = True)
    def prio_func():
        …
```

PyCOMPSs: Task types

- What can be selected as a task?
  - (a) Functions
  - (b) Instance methods
  - (c) Class methods





PyCOMPSs: Main program → Synchronization API

- Data created or updated by a task can be used in the main program of the application
  - But we need to synchronize first!

- Two API methods for synchronization
  - *compss_open* → files

```
my_file = 'file.txt'
func(my_file)
fd = compss_open(my_file)
…
```

func is a task that modifies *my_file*

  - *compss_wait_on* → objects

```
my_obj = MyClass()
my_obj.method()
my_obj = compss_wait_on(my_obj)
…
```

method is a task that modifies *my_obj*

PyCOMPSs: Main program → Future objects

- Mechanism to make asynchronous those tasks that return a value
  - Synchronization is only triggered when necessary
- The future object is a representative of the object yet to be generated

```
@task(returns = MyClass)
def ret_func():
    return MyClass(…)
…
o = ret_func()
```

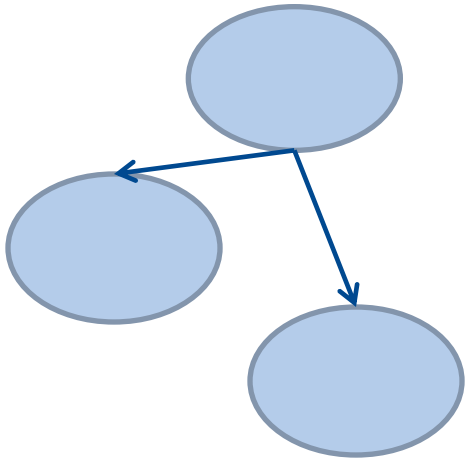
future object

PyCOMPSs: Main program → Future objects (II)

- A future object can be involved in a subsequent task call
  - PyCOMPSs will automatically enforce the dependency

future object ----------•

```
o = ret_func()
…
another_task(o)
…
o.yet_another_task()
```



- Synchronization from main program (same as other objects):

```
o = ret_func()
…
o = compss_wait_on(o)
```

PyCOMPSs: Wrap-up example

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

Main Program

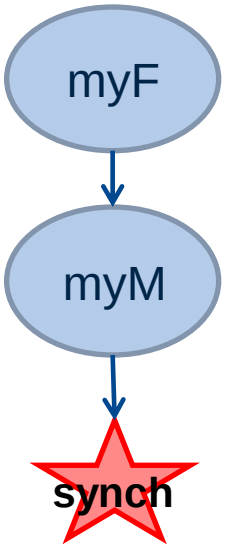
```
foo = Foo()
myFunction(foo)
foo.myMethod()
…
foo = compss_wait_on(foo)
foo.bar()
```

- Task selection in function definition (decorators)

Function definition

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

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



Applications using COMPSs

- Personalized medicine
  - eIMRT: planning radiotherapy treatments
- Earth Science
  - HRT: modeling global ocean-atmosphere circulation
- 3D render
  - LuxRender: renderize architectural designs
- Civil Engineering
  - EnergyPlus: modeling airflows in buildings
  - Architrave: force-effects on buildings

Applications using COMPSs

- BioInformatics
  - Discrete: simulate molecular dynamics for proteins
  - Blast: alignments of protein sequences
  - Hmmer: alignment of protein sequences
  - GeneDetect: genetics algorithm
  - QSAR: drug design

COMPSs Runtime System

Runtime System

Application

Task Selection Interface

Runtime System

Grid

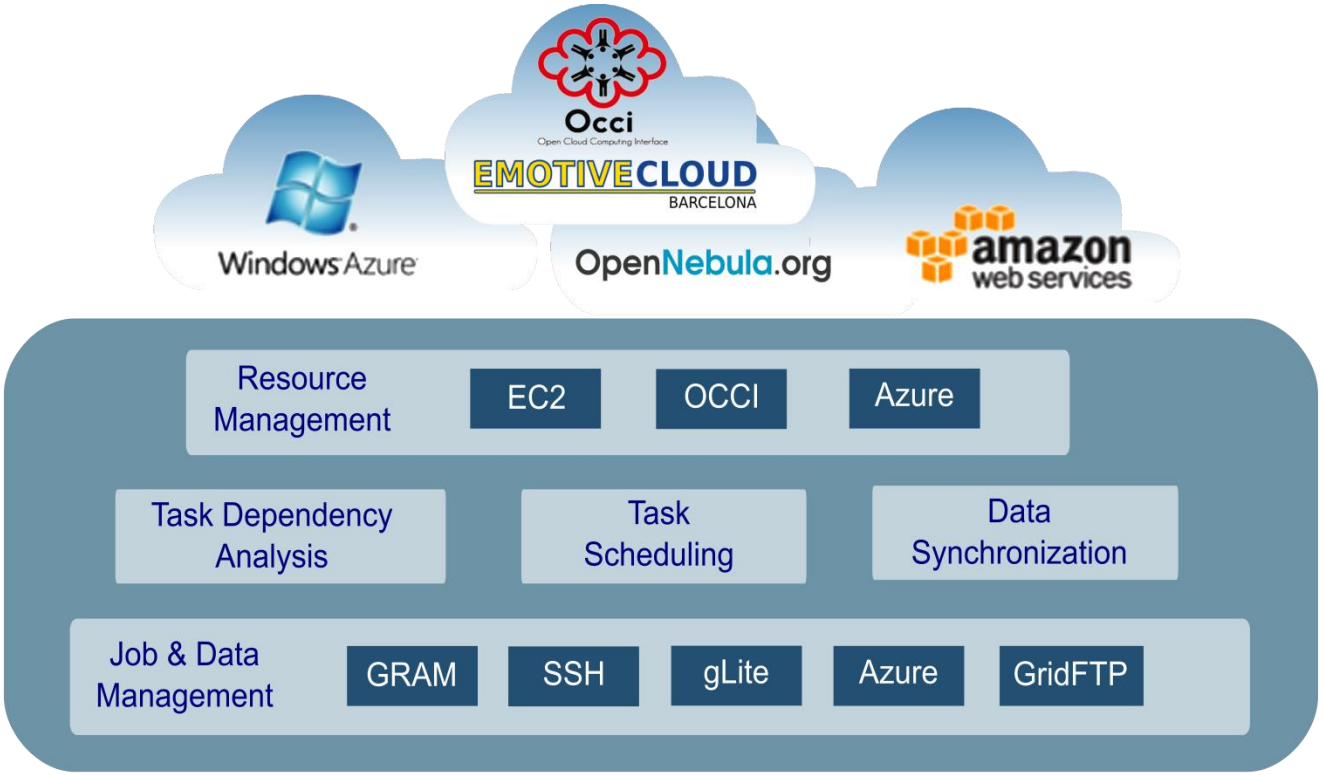
Cluster

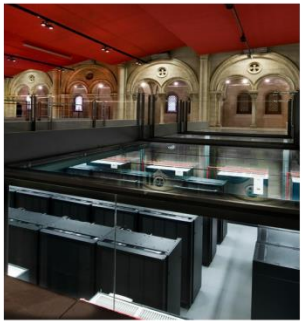
Cloud

Basic features

- Supported Features:
  - Data dependency analysis
  - Data renaming
  - Data transfer
  - Constraint-based scheduling
  - Task versioning support
  - Resource management
  - Results collection
  - Fault tolerance
  - Shared disks support
- In Progress:
  - Checkpointing
  - Task nesting

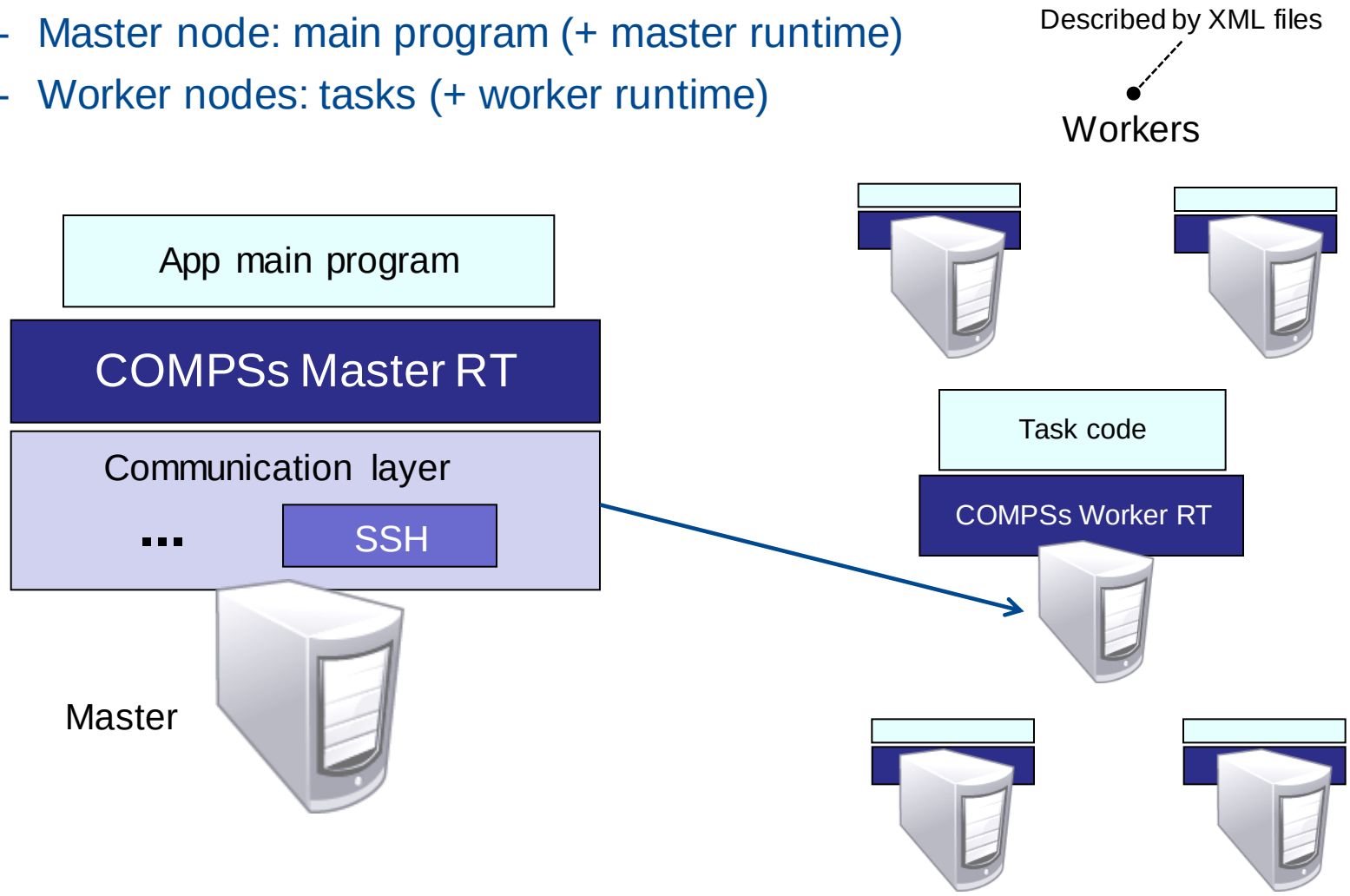
Interoperability





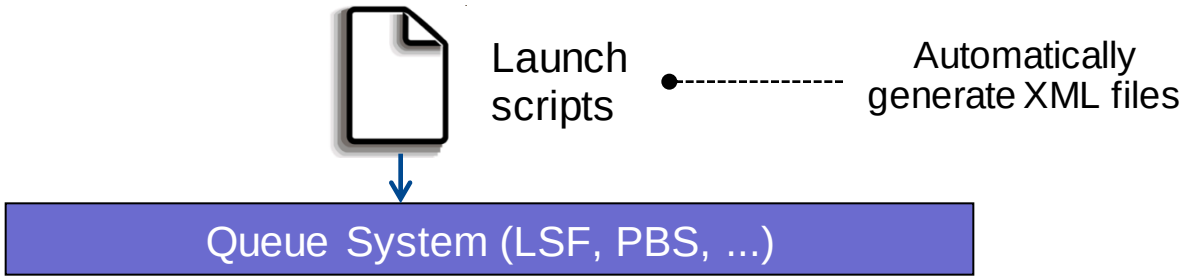
COMPSs in a Cluster (interactive)

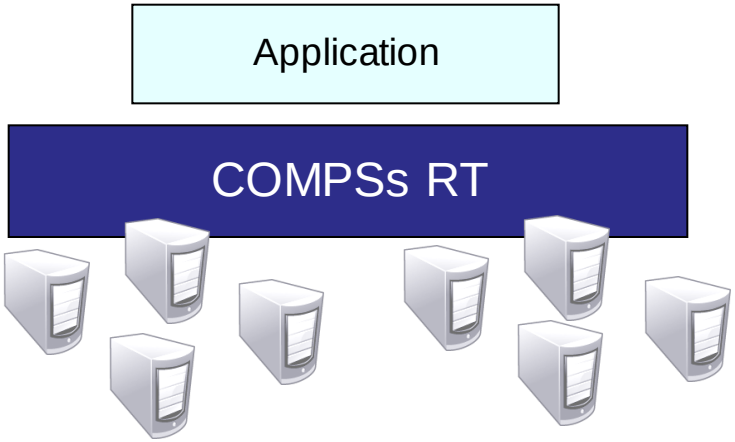
- Typical setup:
  - Master node: main program (+ master runtime)
  - Worker nodes: tasks (+ worker runtime)



COMPSs 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





COMPSs/PyCOMPSs in MNIII

- Environment variables:

```
export IT_HOME=/gpfs/apps/MN3/COMPSs/Trunk/compss/compss-rt/
export GAT_LOCATION=$IT_HOME/../../files/JAVA_GAT
```

- Launching script

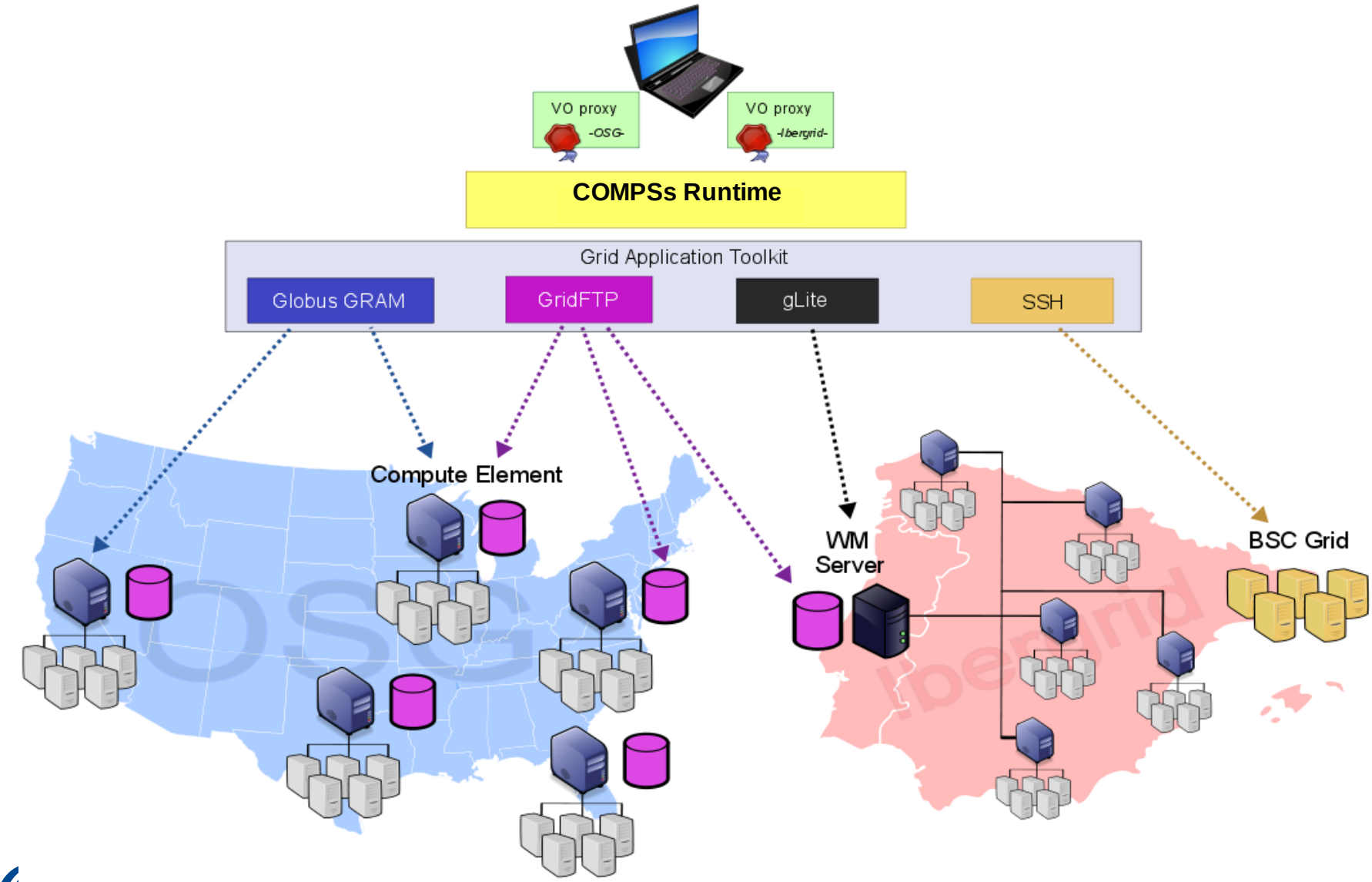
```
$IT_HOME/scripts/queues/run.sh
```

- The script can be executed and prints a help
- Sample execution:

```
$IT_HOME/scripts/queues/run.sh
--app=/gpfs/apps/MN3/COMPSs/Trunk/compss/compss-rt/bindings/python/apps/test/test_mp.py
--exec_time=5 --num_nodes=2
--classpath=/gpfs/apps/MN3/COMPSs/Trunk/compss/compss-rt/bindings/python/apps/
--debug=true --lang=python --tracing=true
```

- Other examples in the application repository

COMPSs in a Grid



Grid/Cluster Configuration: Resources Specification

- Resources.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ResourceList>
    <!--Description for any physical node-->
    <Resource Name="172.20.200.18">
        <Capabilities>
            <Host>
                <TaskCount>0</TaskCount>
                <Queue>short</Queue>
                <Queue/>
            </Host>
            <Processor>
                <Architecture>IA32</Architecture>
                <Speed>3.0</Speed>
                <CoreCount>2</CoreCount>
            </Processor>
            <OS>
                <OSType>Linux</OSType>
                <MaxProcessesPerUser>32</MaxProcessesPerUser>
            </OS>
            <StorageElement>
                <Size>30</Size>
            </StorageElement>
            …
            <Memory>
                <PhysicalSize>1</PhysicalSize>
                <VirtualSize>8</VirtualSize>
            </Memory>
            <ApplicationSoftware>
                <Software>Java</Software>
            </ApplicationSoftware>
            <Service/>
            <VO/>
            <Cluster/>
            <FileSystem/>
            <NetworkAdaptor/>
            <JobPolicy/>
            <AccessControlPolicy/>
        </Capabilities>
        <Requirements/>
    </Resource>
    <Resource Name="172.20.200.19">
        …
    </Resource>
</ResourceList>
```

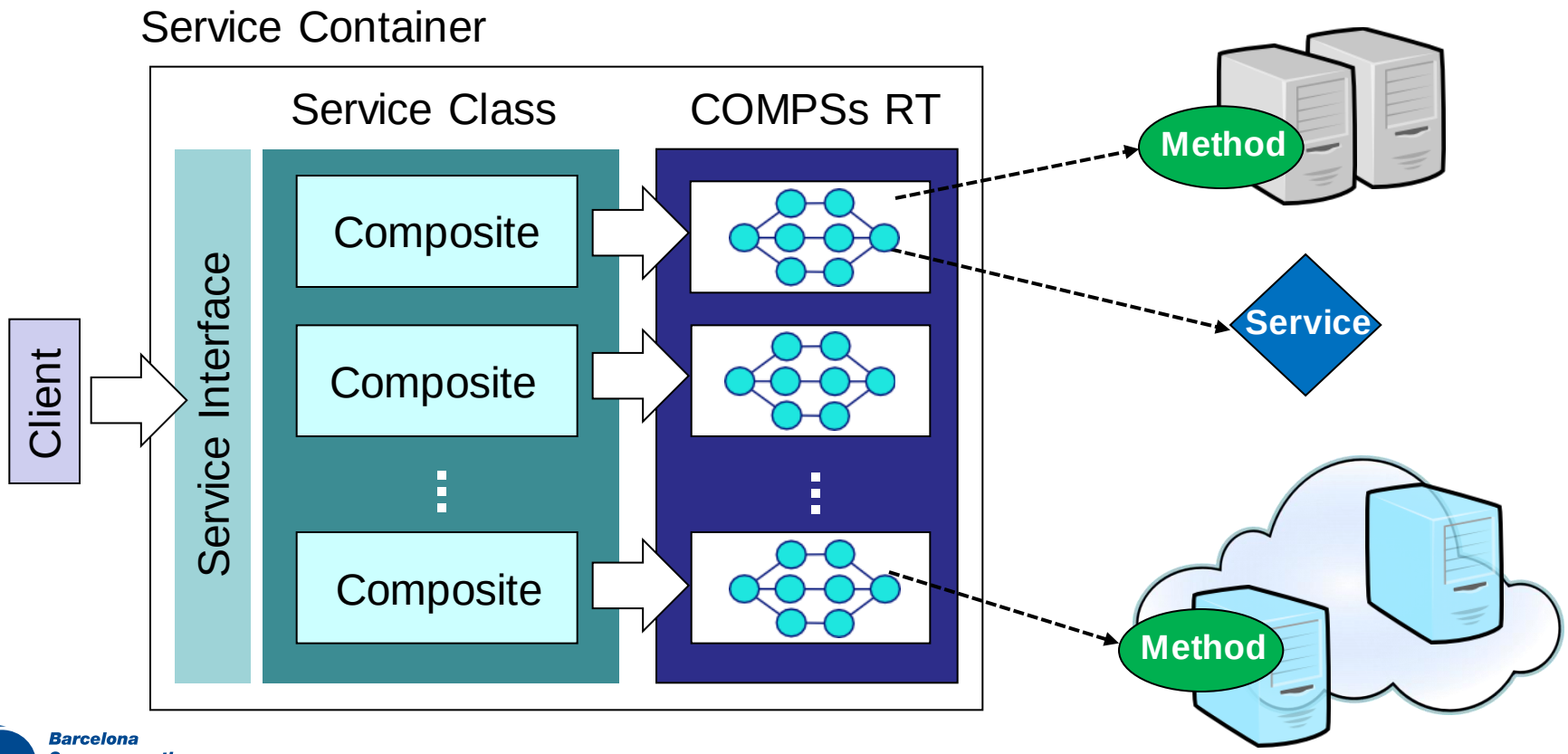
Grid/Cluster Configuration: Project Specification

Project.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Project>
      <!--Description for any physical node-->
      <Worker Name="172.20.200.18">
            <InstallDir>/opt/COMPSs/Runtime/scripts/system/</InstallDir>
            <WorkingDir>/tmp/</WorkingDir>
            <User>user</User>
      </Worker>

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

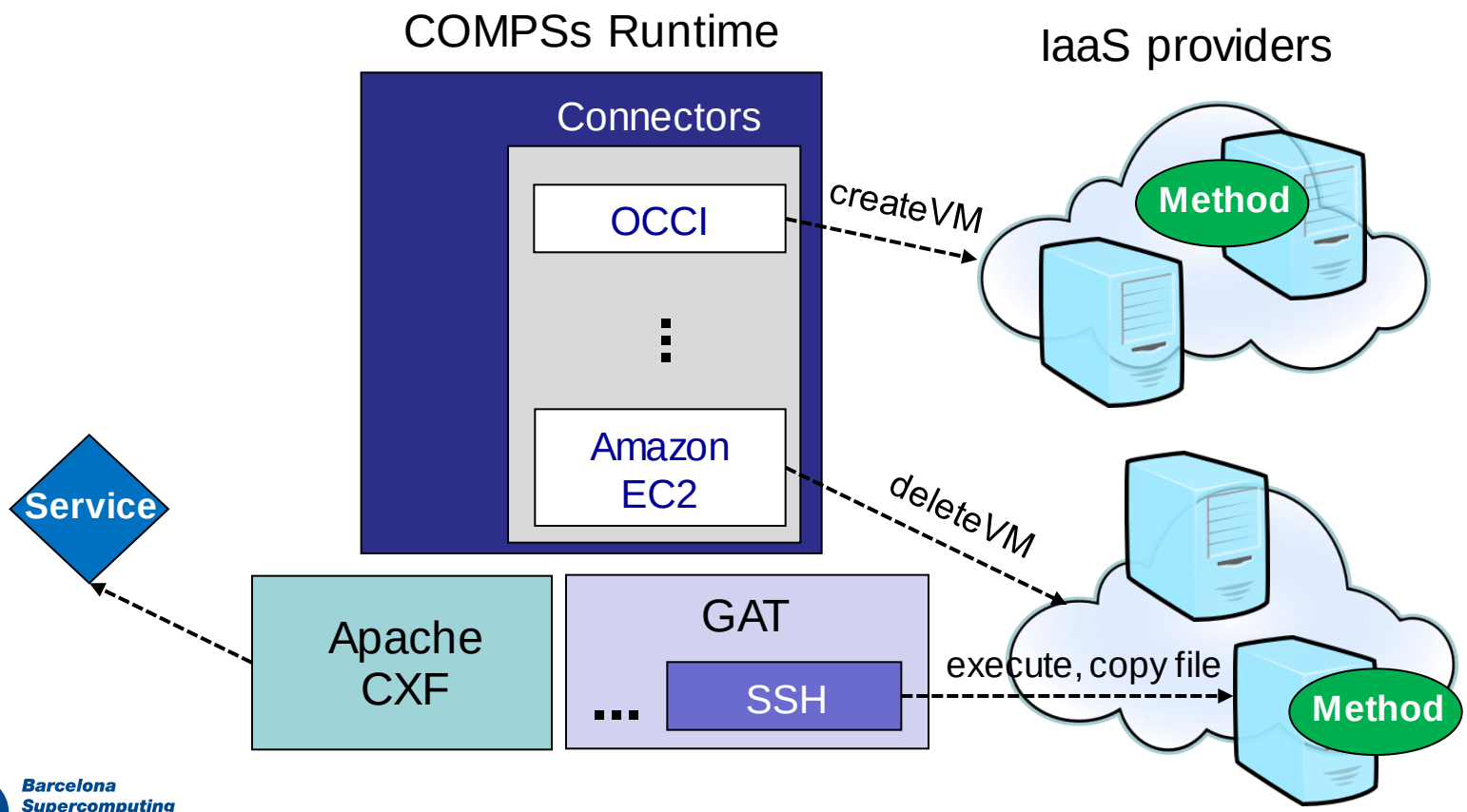
COMPSs in the Cloud

- Runtime integrated in a platform with:
  - Service Oriented Architecture
  - Virtualization



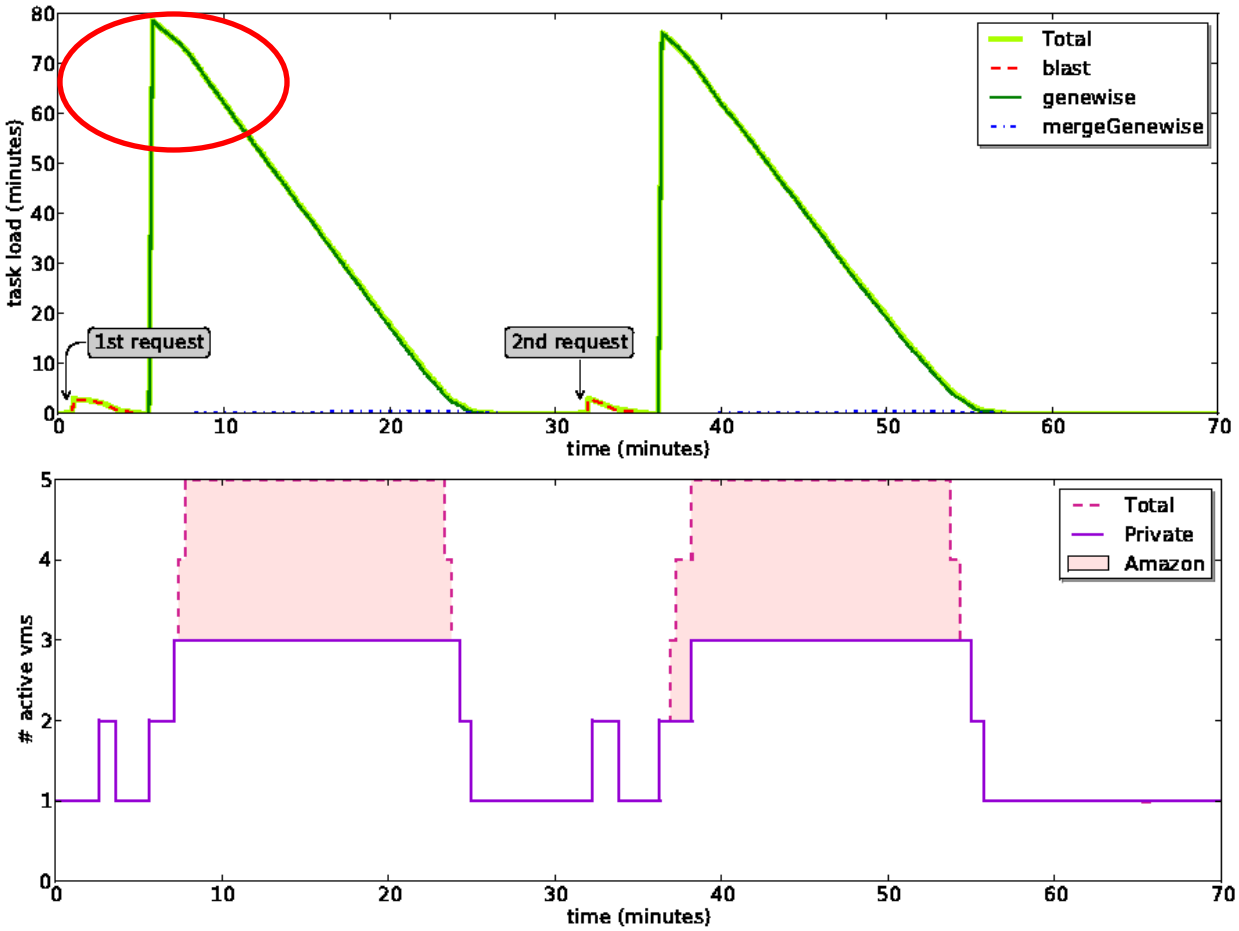
Cloud: Connector design

- Interaction with:
  - Cloud providers: connectors, SSH
  - Service providers: Dynamic WS client



Cloud: Elasticity

- Increase/decrease number of VMs depending on task load
- Bursting to Amazon EC2 to face peak load



Cloud Configuration: Resources Specification

Resources.xml

```
<ResourceList>
  <CloudProvider name="BSCCloud">
      <Server>https://bscgrid20.bsc.es:11443</Server>
      <Connector>integratedtoolkit.connectors.rocci.ROCCI</Connector>
      <ImageList>
          <Image name="debianbase"/>
      </ImageList>
      <CreationTime>120</CreationTime>
      <InstanceTypes>
          <Resource Name="bsc.small">
              <Capabilities>
                  <Processor>
                      <CoreCount>1</CoreCount>
                  </Processor>
                  <StorageElement>
                      <Size>10.0</Size><!– GB -->
                  </StorageElement>
                  <Memory>
                      <PhysicalSize>1</PhysicalSize> ><!– GB -->
                  </Memory>
              </Capabilities>
          </Resource>
          <Resource Name="bsc.medium">
            …
          </Resource>
      </InstanceTypes>
  </CloudProvider>
</ResourceList>
```

Cloud Configuration: Project Specification

Project.xml

```
<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>Owner</Name>
                <Value>userbsc</Value>
            </Property>
    ...
```

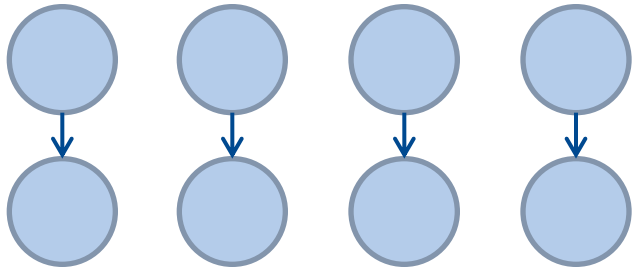
```
...
            <ImageList>
                <Image name="debianbase">
                    <InstallDir>/opt/COMPSs/Runtime/scripts/system/</InstallDir>
                    <WorkingDir>/tmp/</WorkingDir>
                    <User>user</User>
                    <Package>
                        <Source>/home/.../AppName.tar.gz</Source>
                        <Target>/home/user</Target>
                    </Package>
                </Image>
            </ImageList>
            …
            <InstanceTypes>
                <Resource name="bsc.small"/>
            </InstanceTypes>
        </Provider>
    </Cloud>
</Project>
```


BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
DEMOS

Matmul Application

- Application data: matrices
  - Divided in blocks
- Main program:
  - Initialization of the matrices
  - Multiplication of the blocks
- When the app is executed, *multiply* invocations are spawned asynchronously by COMPSs:
  - Data dependencies between tasks are automatically managed
  - Tasks are scheduled on available distributed resources when free of dependencies
  - Task data is transferred when necessary

Task dependency graph for matrices of 2x2 blocks (dynamically generated)



Matmul: User code

```
for (int i = 0; i < MSIZE; i++){
        for (int j = 0; j < MSIZE; j++){
                for (int k = 0; k < MSIZE; k++){
                        long ini, fi;
                        ini = System.currentTimeMillis();
                        MatmulImpl.multiplyAccumulative( _C[i][j], _A[i][k], _B[k][j] );
                        fi = System.currentTimeMillis();
                        System.out.println("TASK: " + ((fi - ini) / 1000) + " seconds\n");
                }
        }
}
```

```
public static void multiplyAccumulative( String f3, String f1, String f2 )
{
        Block a = new Block( f1 );
        Block b = new Block( f2 );
        Block c = new Block( f3 );
        c.multiplyAccum( a, b );
        try
                ...
}

public void multiplyAccum ( Block a, Block b )
{
        for( int i = 0; i < this.bRows; i++ )                   // rows
                for( int j = 0; j < this.bCols; j++ )           // cols
                        for ( int k = 0; k < this.bCols; k++ )  // cols
                                this.data[i][j] += a.data[i][k] * b.data[k][j];
}
```

Matmul: interface

```
package matmul;

import integratedtoolkit.types.annotations.Constraints;
import integratedtoolkit.types.annotations.Method;
import integratedtoolkit.types.annotations.Parameter;
import integratedtoolkit.types.annotations.Parameter.*;


public interface MatmulItf {
@Constraints(processorCoreCount = 4, memoryPhysicalSize = 1.5f)
@Method(declaringClass = "matmul.MatmulImpl")
void multiplyAccumulative(
        @Parameter(type = Type.FILE, direction = Direction.INOUT)
        String file1,

        @Parameter(type = Type.FILE, direction = Direction.IN)
        String file2,

        @Parameter(type = Type.FILE, direction = Direction.IN)
        String file3
);

}
```

Matmul: Compiling

- Compiling with command line:
  - cd workspace/matmul/
  - javac src/main/java/matmul/*.java
  - cd src/main/java/
  - jar cf matmul.jar matmul

- From eclipse:
  - Package Explorer -> Project (matmul) -> Export…

Matmul: Deploying

- In this case, in the same machine
  - Copy to home directory
  - cp matmul.jar ~
  - cd

- In remote machines
  - Code needs to be transfer to machine that will host main code

Matmul: Monitoring

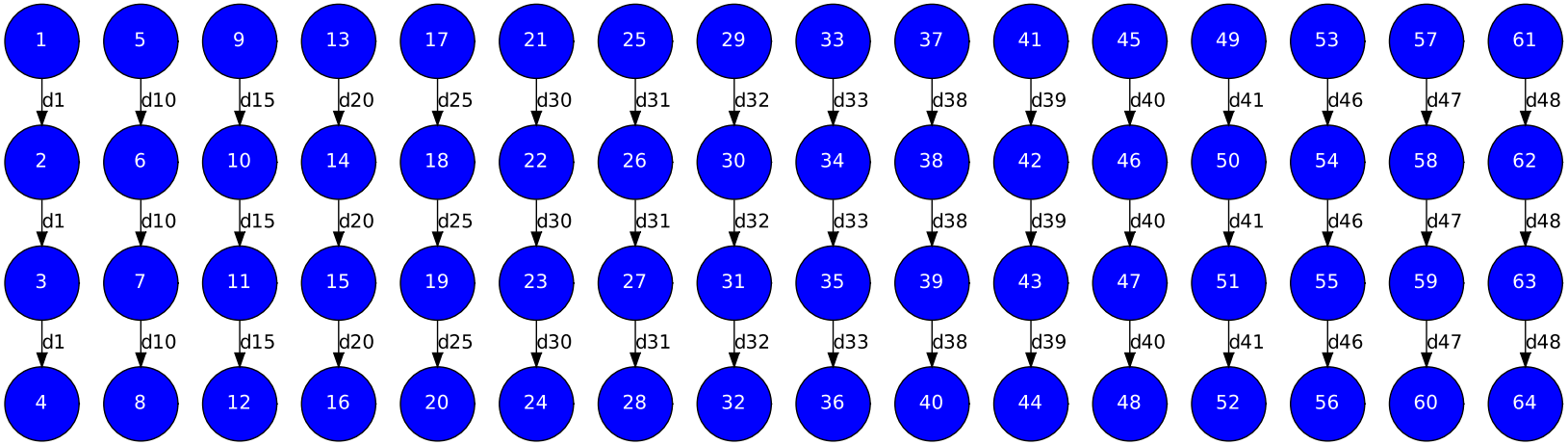
- Browse
  - http://localhost:8080/compss-monitor

Matmul: Executing

- Set CLASSPATH
  - export CLASSPATH=$CLASSPATH:/home/user/matmul.jar
  - runcompss matmul.Matmul 8

Demos: Matmul

- Blocks matrix multiplication



Matmul Python

- Task selection: function *multiply*
  - Multiplies two blocks
  - Parameters:
    - fa, fb: file names for input blocks A and B (FILE type, IN direction)
    - fc: file name for input-output block C (FILE type, INOUT direction)
    - size: block size (input integer, type and direction automatically inferred)

Matmul with PyCOMPSs

Main program

```
from pycompss.api.api import compss_open
import sys

args = sys.argv[1:]

MSIZE = int(args[0])
BSIZE = int(args[1])

A = []
B = []
C = []

initialize_variables()
fill_matrices()

for i in range(MSIZE):
    for j in range(MSIZE):
        for k in range(MSIZE):
            multiply(A[i][k], B[k][j], C[i][j], BSIZE)

for i in range(MSIZE):
    for j in range(MSIZE):
        fd = compss_open(C[i],[j], 'r')
        print_block(fd)
```

Task

```
from pycompss.api.task import task
from pycompss.api.parameter import *

@task( fa = FILE, fb = FILE, fc = FILE_INOUT)
def multiply(fa, fb, fc, size):
    a = load_block(fa)
    b = load_block(fb)
    c = load_block(fc)
    for i in range(size):
        for j in range(size):
            for k in range(size):
                c[i][j] += a[i][k] * b[k][j];
    store_block(c, fc, size)
```

Matmul Python: Execution

- **Execute the application:**

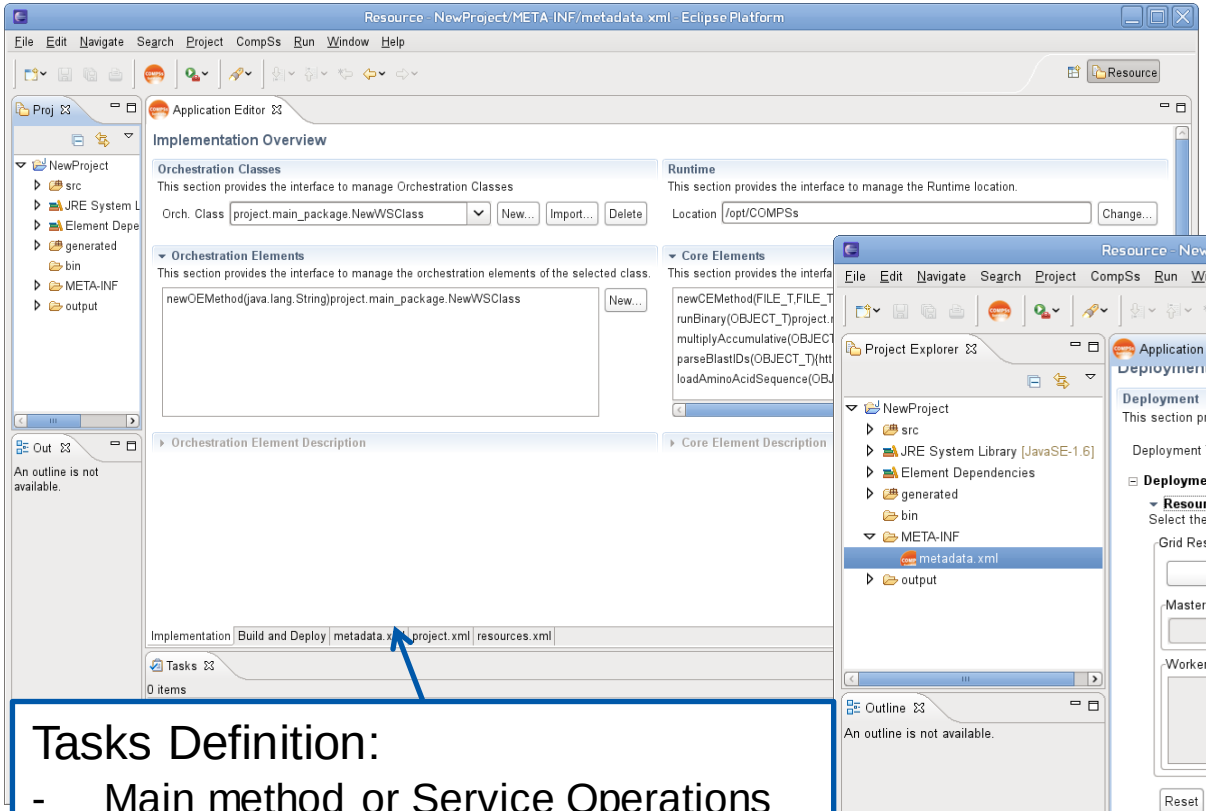
```
runcompssext --lang=python \
--app=/home/user/python_workspace/matmul-python/matmul/matmul.py \
--classpath=/home/user/python_workspace/matmul-python/ \
--cline_args="8 64"
```

- **For available command options**:

```
runcompssext --help
```

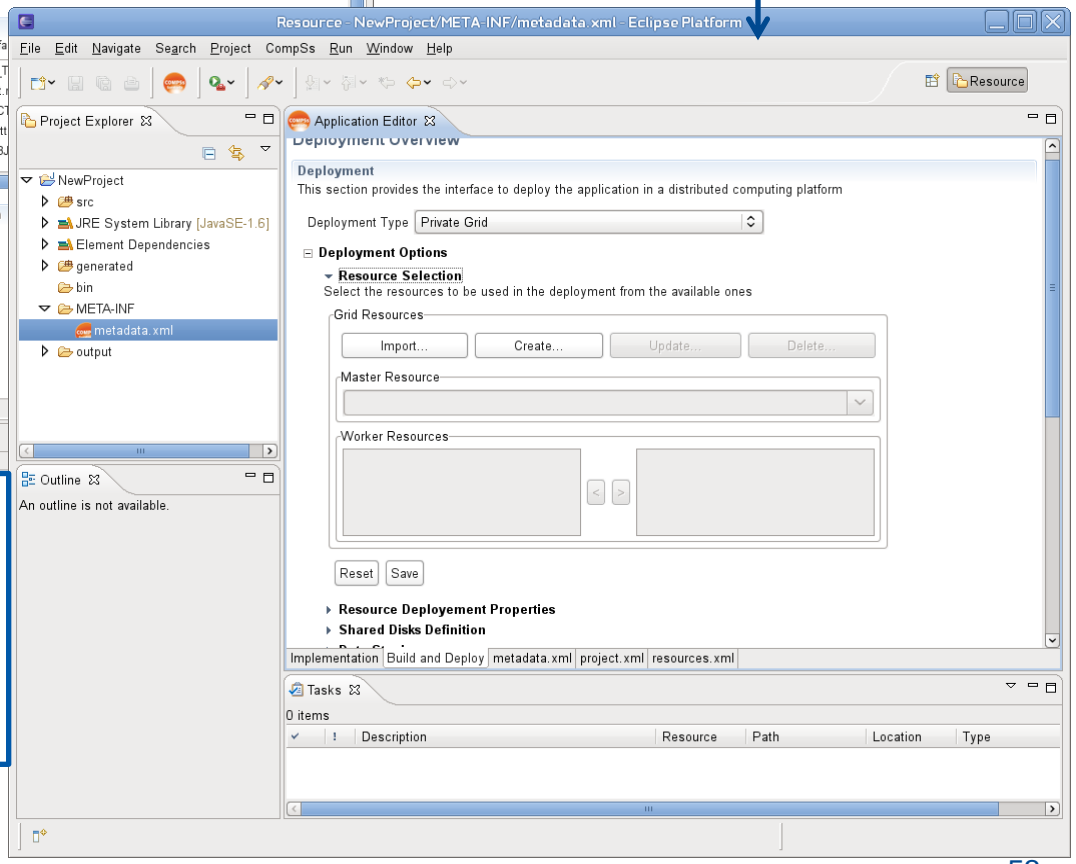
IDE COMPSs applications

IDE for implementing and deploying applications



Building & Deployment:

- Generate Packages
- Define hosts & Deploy



Tasks Definition:

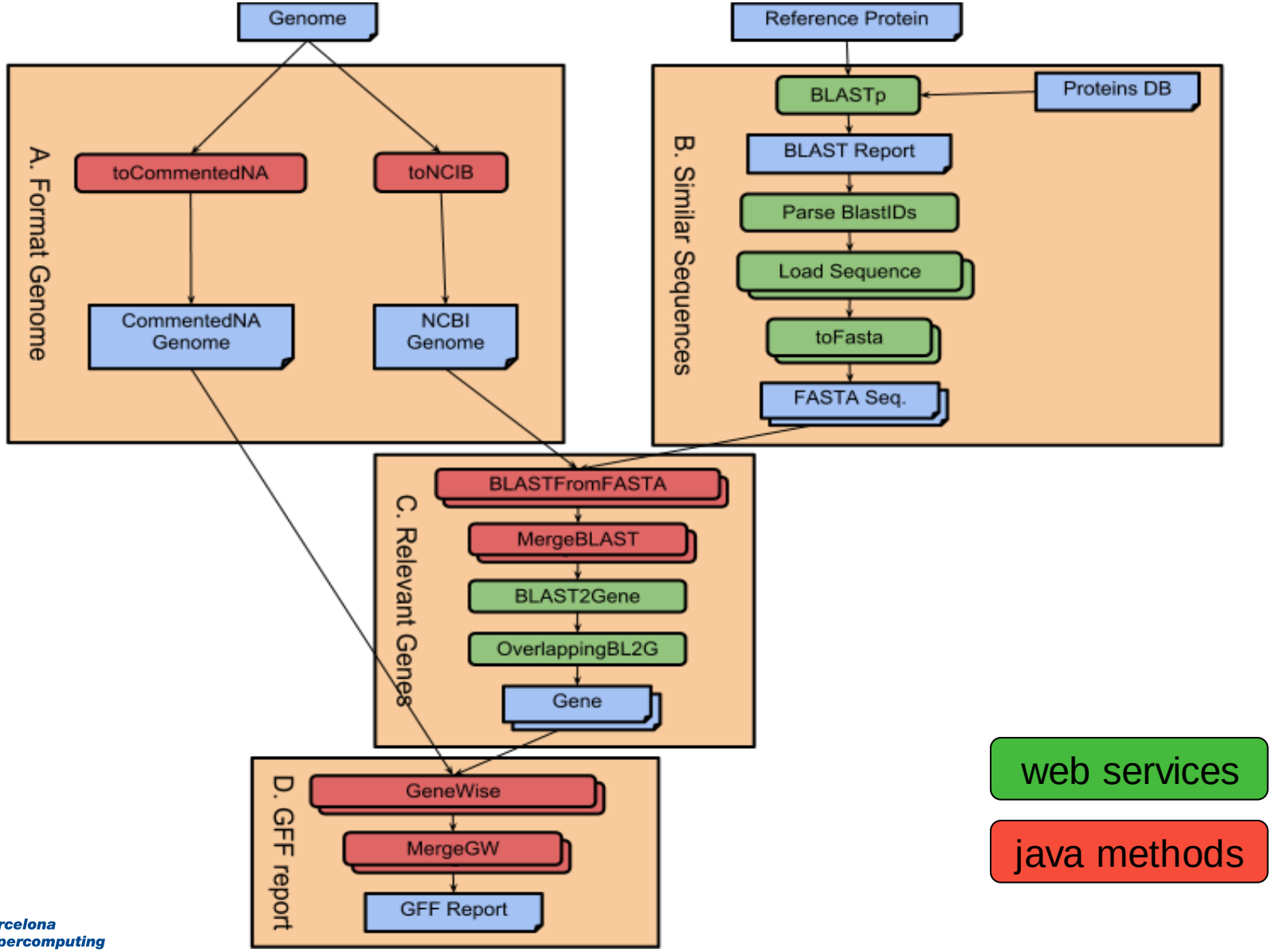
- Main method or Service Operations (Orchestration Elements)
- Java Methods and Web Services (Core Element)

Demos: Gene Detection Application

- Gene Detection algorithm designed by the BSC Life Sciences department
  - Automatic Homology-based gene detection and analysis

- Combine services with computations
  - Example that shows different capabilities of COMPSs
  - Implicit synchronization points
  - Different method and service call types
  - Objects and files as parameters

Gene Detection




BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
HANDS-ON

Hands-On: Overview

- COMPSs Virtual Machine Set-up
- Java Application
  - Code Modification
  - Configuration, Compilation & Execution
  - Monitoring, debugging, tracing
- Python Application
- IDE

COMPSs development VM Installation

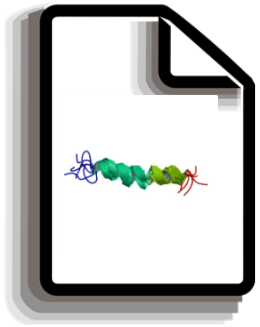
- COMPSs Virtual Appliance
  - Available from website
    - http://compss.bsc.es/releases/vms/compss-latest-vm.ova
- VirtualBox: Import Virtual Appliance…

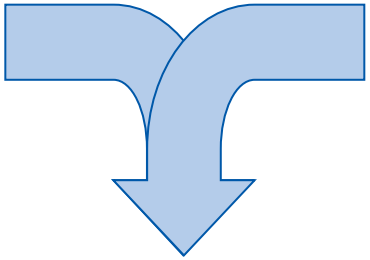
Java Hands-on

Application: HMMER suite (hmmpfam)

- hmmpfam is part of the HMMER suite: set of tools for protein sequence analysis
  - Reads a sequences file and compares each sequence in it against a database of HMMs
  - HMM (Hidden Markov Model): statistical figure that represents a protein family
- Goal: create an hmmpfam efficient implementation
  - Starting point: sequential version of the hmmpfam tool

Protein Database



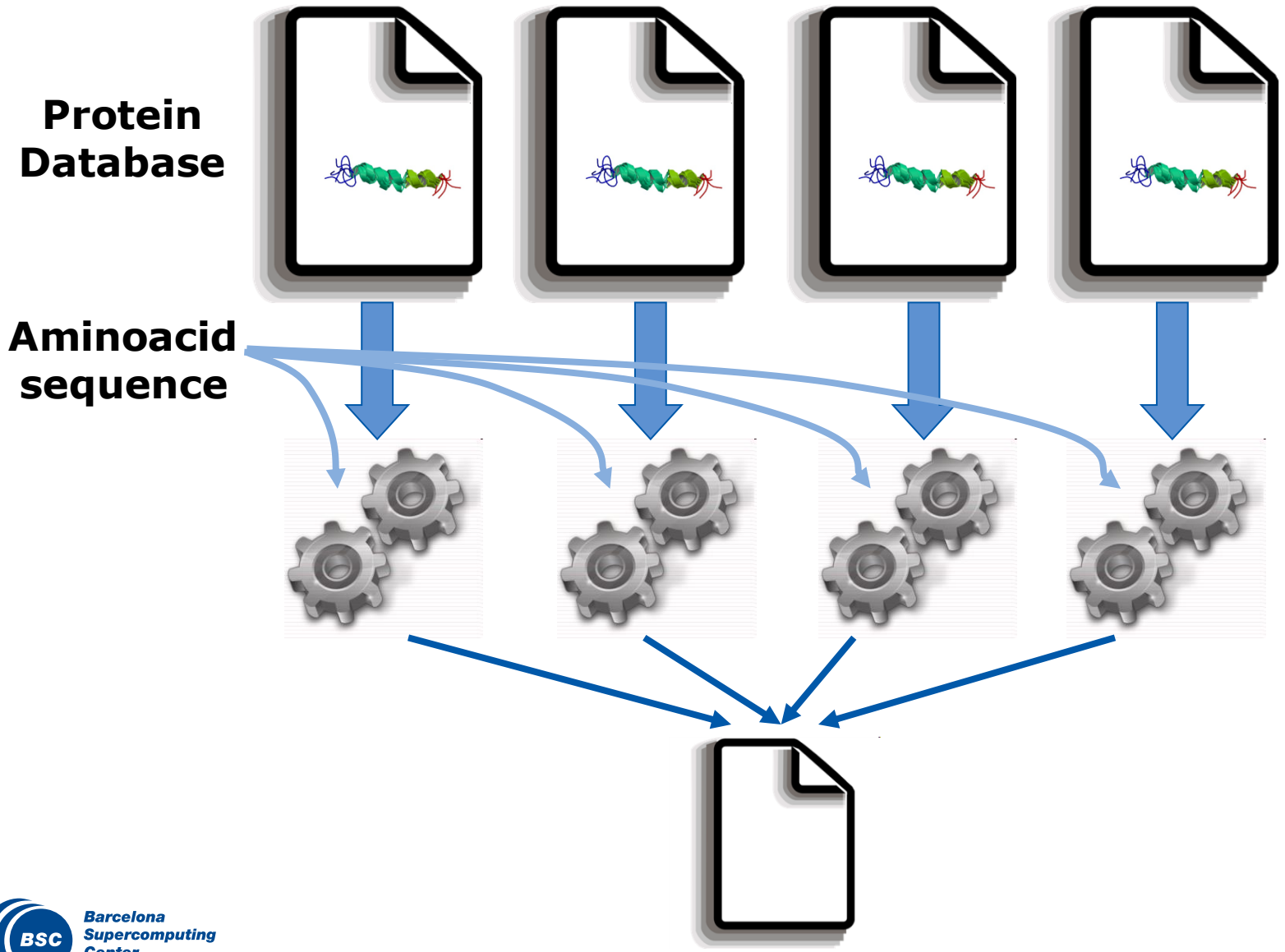


Aminoacid Sequence

IQKKSGKWHTLTDLRAVNAVI
QPMGPLQPGLPSPAMIPKDW
PLIIIDLKDCFFTIPLAEQDCEK
FAFTIPAINNKEPATRF

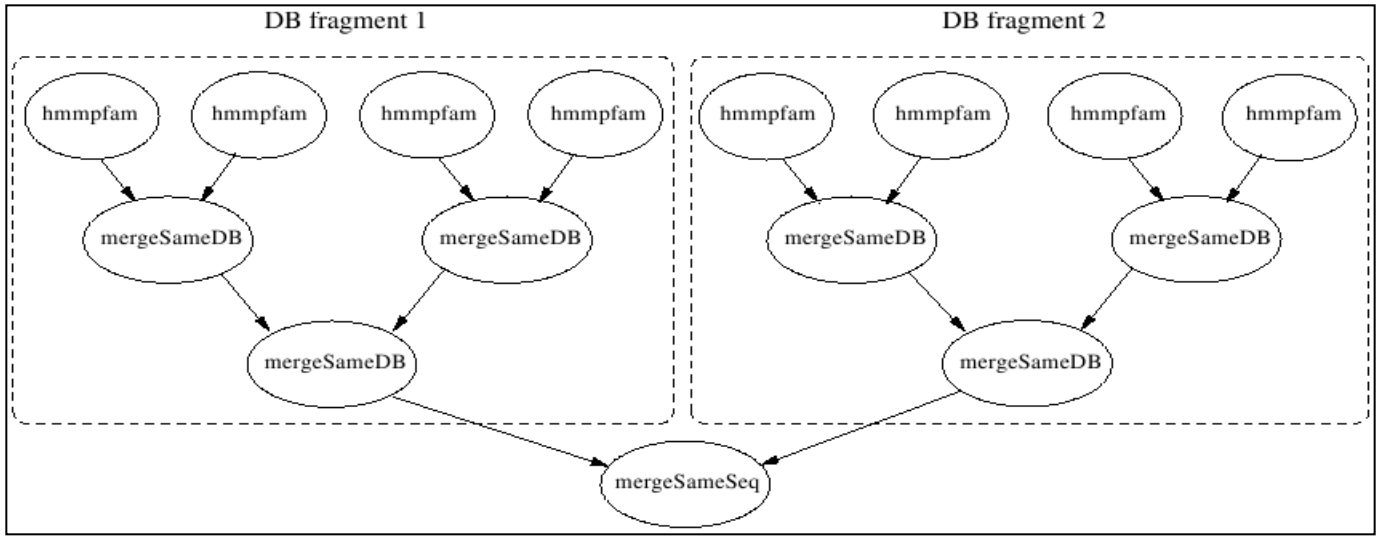
Model	Score	E-value	N
IL6_2	-78.5	0.13	1
COLFI_2	-164.5	0.35	1
pgtp_13	-36.3	0.48	1
clf2	-15.6	3.6	1
PKD_9	-24.0	5	1

HMMER Hands On: Parallelization



HMMER Hands on Parallelization with COMPSs

- With the COMPSs: hmmpfam becomes parallel
  - Phase 1: Split both input sequences and database
  - Phase 2: Process them in parallel (speed up execution)
  - Phase 3: Reduction of results
    - mergeSameDB: merge the results of the DB fragment
    - mergeSameSeq: merge the different DB fragments



HMMER Hands On: Exercise

- Complete the HMMER parallelization with COMPSs
  - Define mergeSameDB method in the interface.
  - Complete the part of the main code to do the first merge using this method

HMMER Hands On: Exercise Solution

```
for (…){
…

    String[] sequences = outputs[db];
    while (neighbor < numSeqFrags) {
        for (int seq = 0; seq < numSeqFrags; seq += 2 * neighbor) {
            if (seq + neighbor < numSeqFrags) {
                sequences[seq] = HMMPfamImpl.mergeSameDB(sequences[seq],
                                                         sequences[seq + neighbor]);
            }
        }
        neighbor *= 2;
    }
}
```

HMMER Hands On: Exercise Solution

```
public interface HMMPfamItf {

    …

    @Method(declaringClass = "hmmerobj.HMMPfamImpl")
    void mergeSameDB(
            @Parameter(type = Type.OBJECT, direction = Direction.IN)
            String result1,
            @Parameter(type = Type.OBJECT, direction = Direction.IN)
            String result2
    );
    …
}
```

Implementation

Parameter metadata

HMMER: Configuration Compilation and Execution

- Project.xml: /opt/COMPSs/Runtime/xml/projects/project.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Project>
      <!--Description  for any physical  node-->
      <Worker Name="localhost">
            <InstallDir>/opt/COMPSs/Runtime/scripts/system/</InstallDir>
            <WorkingDir>/tmp/</WorkingDir>
            <User>user</User>
      </Worker>

      ...
</Project>
```

HMMER: Configuration Compilation and Execution

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

```
<?xml version="1.0" encoding="UTF-8"?>
<ResourceList>
    <!--Description for any physical node-->
    <Resource Name="localhost">
        <Capabilities>
            <Host>
                <TaskCount>0</TaskCount>
                <Queue>short</Queue>
                <Queue/>
            </Host>
            <Processor>
                <Architecture>AMD64</Architecture>
                <Speed>3.0</Speed>
                <CoreCount>4</CoreCount>
            </Processor>
            <OS>
                <OSType>Linux</OSType>
                <MaxProcessesPerUser>32</MaxProcessesPerU
            </OS>
            <StorageElement>
                <Size>30</Size>
            </StorageElement>
            …
            …
            <Memory>
                <PhysicalSize>2</PhysicalSize>
                <VirtualSize>8</VirtualSize>
            </Memory>
            <ApplicationSoftware>
                <Software>Java</Software>
            </ApplicationSoftware>
            <Service/>
            <VO/>
            <Cluster/>
            <FileSystem/>
            <NetworkAdaptor/>
            <JobPolicy/>
            <AccessControlPolicy/>
        </Capabilities>
        <Requirements/>
    </Resource>

<ResourceList>
```

HMMER: Configuration Compilation and Execution

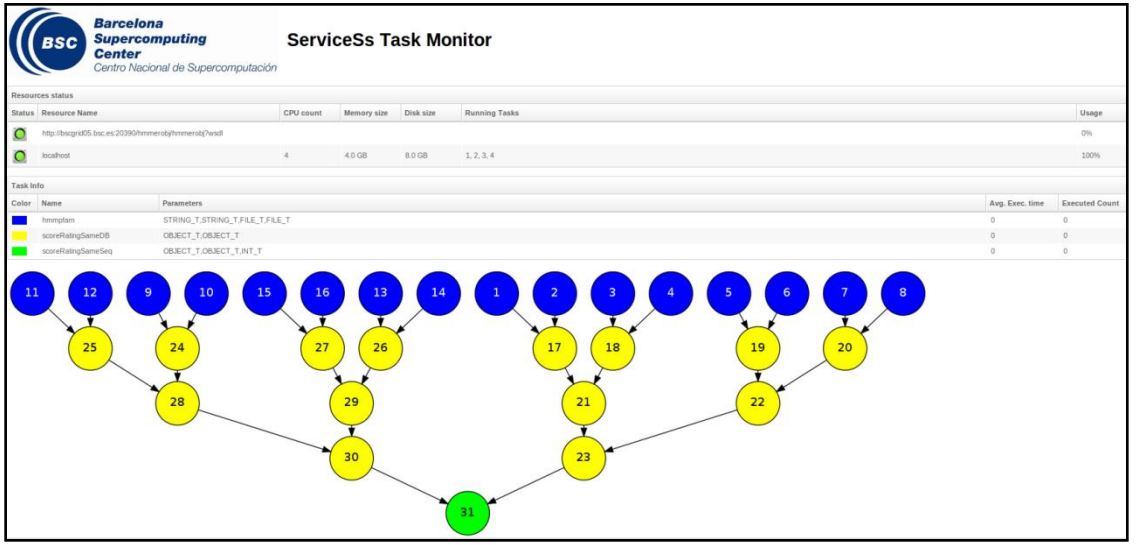
- Compilation (Eclipse IDE)
  - Package Explorer -> Project (hmmerobjblanks) -> Export… (Hands-on)
  - Package Explorer -> Project (hmmerobj) -> Export… (Solution)
- Usage
  - runcompss hmmerobj.HMMPfam <database> <sequences> <output> <params>
- Execution
  - cp ~/workspace/hmmerobj/jar/hmmerobj.jar ~
  - export CLASSPATH=$CLASSPATH:/home/user/hmmerobj.jar
  - runcompss hmmerobj.HMMPfam /sharedDisk/Hmmer/smart.HMMs.bin /sharedDisk/Hmmer/256seq /home/user/out.txt 2 8 -A 222

HMMER: Configuration Compilation and Execution

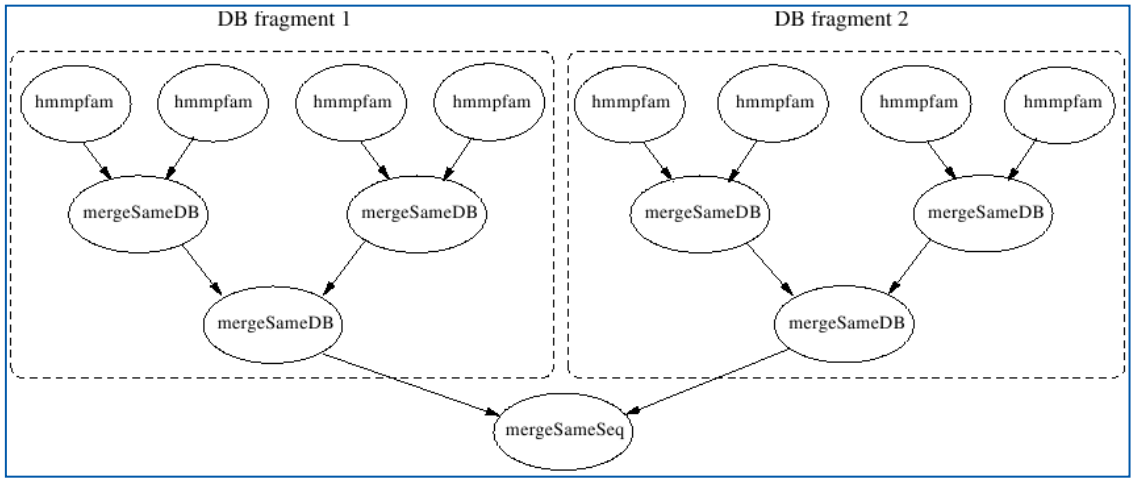
```
user@bsccompss:~$ runcompss hmmerobj.HMMPfam /sharedDisk/Hmmer/smart.HMMs.bin /sharedDisk/Hmmer/256seq
      /home/user/out.txt 2 8 -A 222
-e
---------- Executing hmmerobj.HMMPfam in IT mode total----------

[ API] - Deploying the Integrated Toolkit
[ API] - Starting the Integrated Toolkit
[ API] - Initializing components
[ API] - Ready to process tasks
[ API] - Opening file /tmp/hmmer_frags/seqF0_1 in mode WRITE
[ API] - Opening file /tmp/hmmer_frags/seqF1_1 in mode WRITE
[ API] - Opening file /tmp/hmmer_frags/seqF2_1 in mode WRITE
[ API] - Opening file /tmp/hmmer_frags/seqF3_1 in mode WRITE
[ API] - Opening file /tmp/hmmer_frags/seqF4_1 in mode WRITE
[ API] - Opening file /tmp/hmmer_frags/seqF5_1 in mode WRITE
[ API] - Opening file /tmp/hmmer_frags/seqF6_1 in mode WRITE
[ API] - Opening file /tmp/hmmer_frags/seqF7_1 in mode WRITE
[ API] - Opening file /tmp/hmmer_frags/dbF0_1 in mode WRITE
[ API] - Opening file /tmp/hmmer_frags/dbF1 in mode WRITE
[ API] - Opening file /home/user/out.txt in mode WRITE
[ API] - No more tasks for app 1
[ API] - Stopping IT
[ API] - Cleaning
[ API] - Integrated Toolkit stopped

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

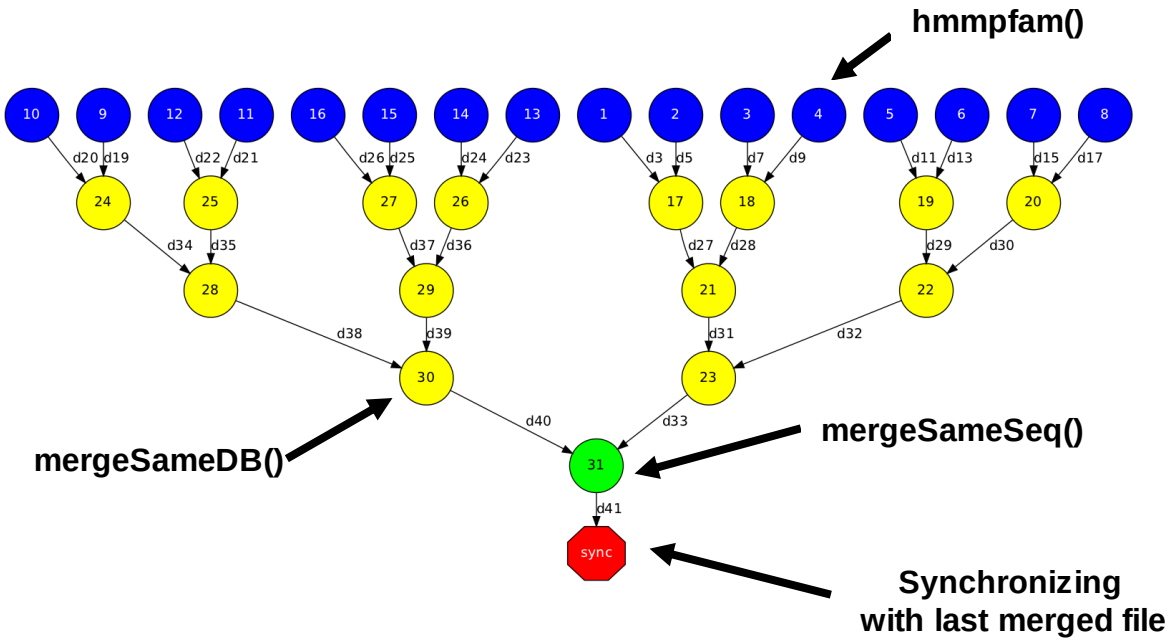
HMMER Hands On: Monitoring

- The runtime of COMPSs provides real-time monitoring
  - http://localhost:8080/compss-monitor/
- The user can follow the progress of the application:
  - # tasks
  - Resources usage information
  - Execution time per task
  - Real-time execution graph
  - Etc.



HMMER Hands On: Workflow





HMMER Hands On: Debugging

- COMPSs can be run in debug mode showing more information about the execution allowing to detect possible problems
  - Log level configurable at: /opt/COMPSs/Runtime/log/it-log4j

- The user can check the execution of its application by reading:
  - The output/errors of the main application (console stdout)
  - The output/error of a task # N
    - $HOME/IT/[APP_NAME]/jobs/jobN.[out|err]
  - Messages from the runtime COMPSs
    - $HOME/it.log

- The user can verify the correct structure of the parallel application generating a complete post-mortem application graph
  - gengraph $HOME/APP_NAME.dot

HMMER Hands On: Tracing (Overview)

- 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

HMMER Hands On: Tracing (Instrumentation)

- 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
- For more information about Extrae visit:
  - http://www.bsc.es/computer-sciences/extrae

HMMER Hands On: Tracing (Instrumentation)

- 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
- For more information about Extrae visit:
  - http://www.bsc.es/computer-sciences/extrae

HMMER Hands On: Tracing (Instrumentation)

---------- Executing hmmerobj.HMMPfam ----------

[API] - Deploying the Integrated Toolkit
[API] - Starting the Integrated Toolkit
[API] - Initializing components

COMPSs runtime starts

Welcome to Extrae 2.4.3rc4 (revision 311 based on framework/trunk/files/extrae)
Extrae: Generating intermediate files for Paraver traces.
Extrae: Intermediate files will be stored in /home/user/IT/hmmerobj.HMMPfam
Extrae: Tracing buffer can hold 500000 events
Extrae: Tracing mode is set to: Detail.
Extrae: Successfully initiated with 1 tasks

Extrae starts before the user application execution

[API] - Ready to process tasks

…
…
…

Extrae keeps tracing events in background

HMMER Hands On: Tracing (Instrumentation)

[API] - No more tasks for app 1
[API] - Stopping IT
[API] - Cleaning
Extrae: Application has ended. Tracing has been terminated.
…

The application finishes and the tracing process ends

The merge process starts

merger: Output trace format is: Paraver
merger: Extrae 2.4.3rc4 (revision 311 based on framework/trunk/files/extrae)
…

[API] - Integrated Toolkit stopped
…

COMPSs runtime ends

Intermediate trace files are processed

mpi2prv: Selected output trace format is Paraver

mpi2prv: Parsing intermediate files

The final trace file is generated

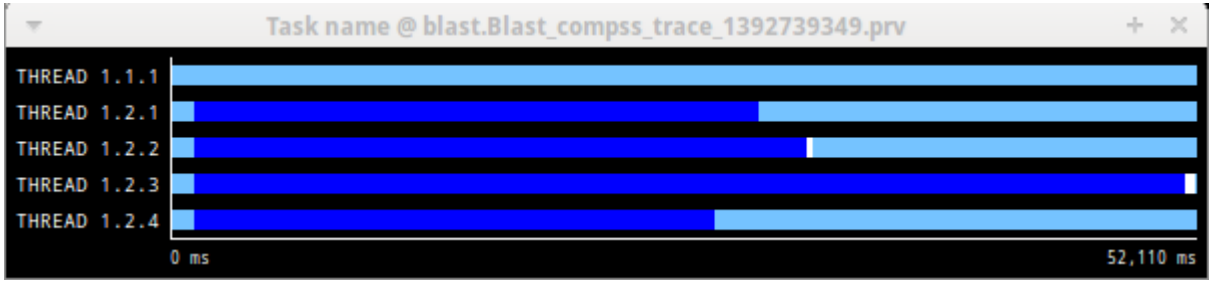
mpi2prv: Generating tracefile (intermediate buffers of 1342156 events)

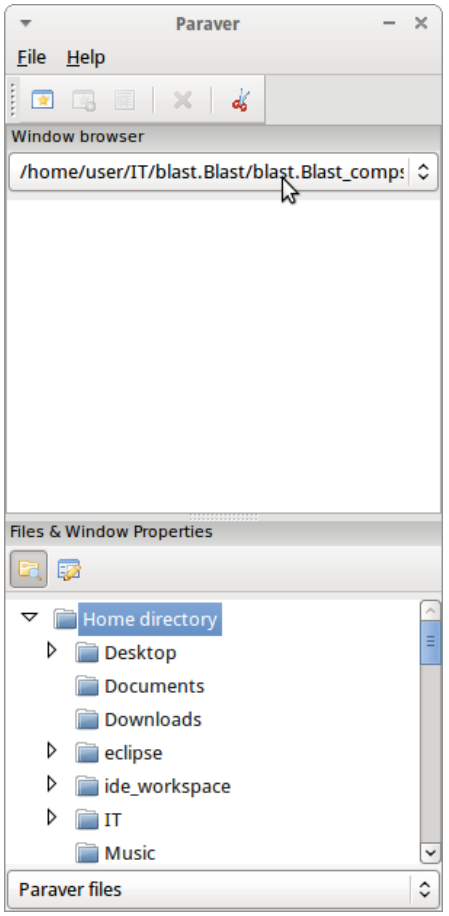
mpi2prv: Congratulations! hmmerobj.HMMPfam_compss_trace_1392736225.prv has been generated.

HMMER Hands On: Tracing (Visualization)

- 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:
  - http://www.bsc.es/computer-sciences/performance-tools/paraver



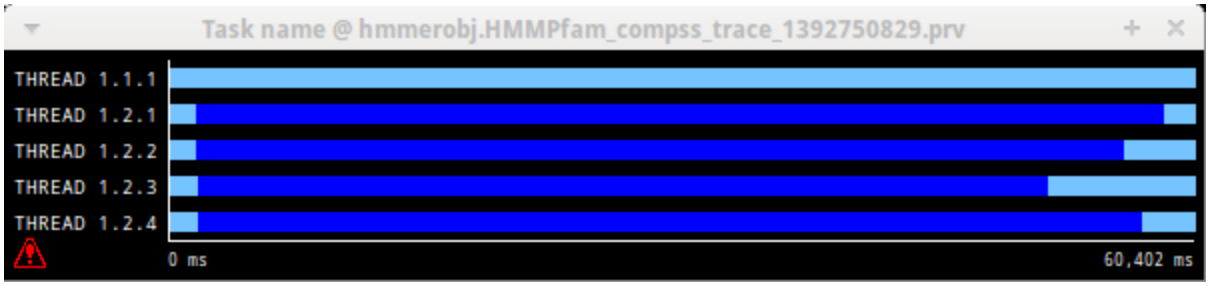


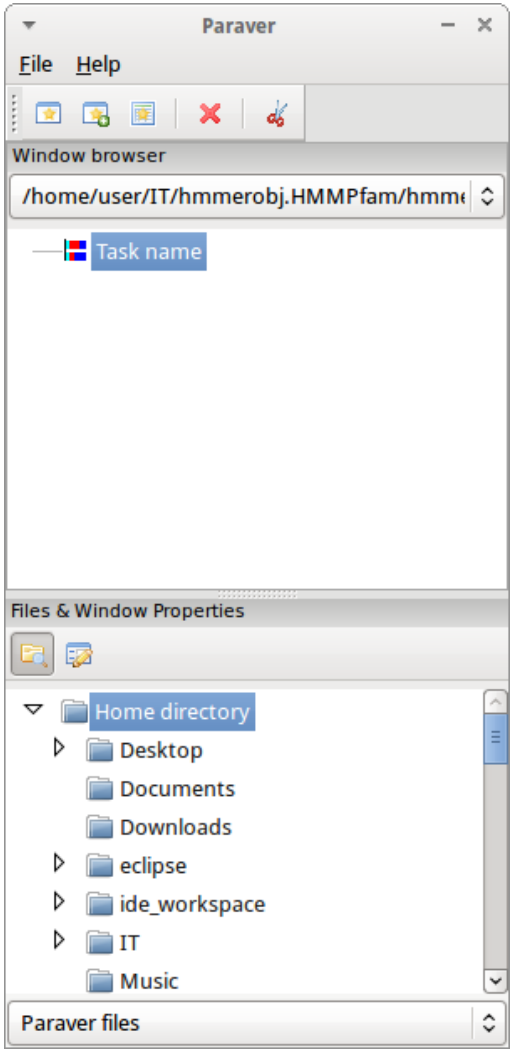
HMMER Hands On: Tracing

- Compilation (Eclipse IDE)
  - *Package Explorer -> Project (hmmerobj) -> Export…*
- Execution
  - *cp $HOME/workspace/hmmerobj/jar/hmmerobj.jar /home/user*
  - *export CLASSPATH=$CLASSPATH:/home/user/hmmerobj.jar*
  - *runcompssext --app=hmmerobj.HMMPfam --tracing=true --cline_args="/sharedDisk/Hmmer/smart.HMMs.bin /sharedDisk/Hmmer/256seq /home/user/out.txt 2 8 -A 222“*
- Open paraver
  - *wxparaver /home/user/IT/hmmerobj.HMMPfam/*.prv*

HMMER Hands On: Tracing

- COMPSs provides a configuration file to automatically obtain the view of the trace
  - File / Load Configuration…
  - /opt/COMPSs/paraver/cfgs/tasks.cfg

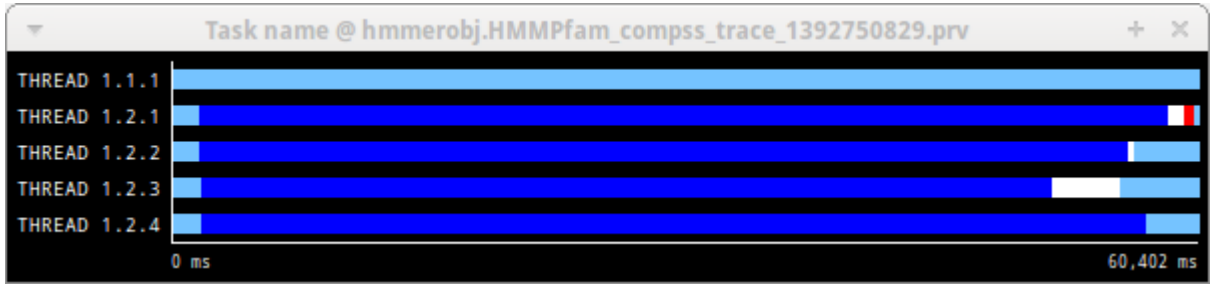




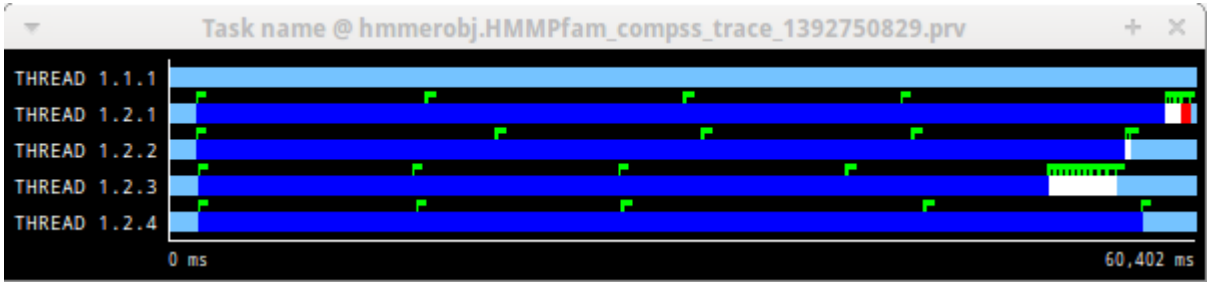
- Some small adjustments must be done in order to view the trace correctly

HMMER Hands On: Tracing

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

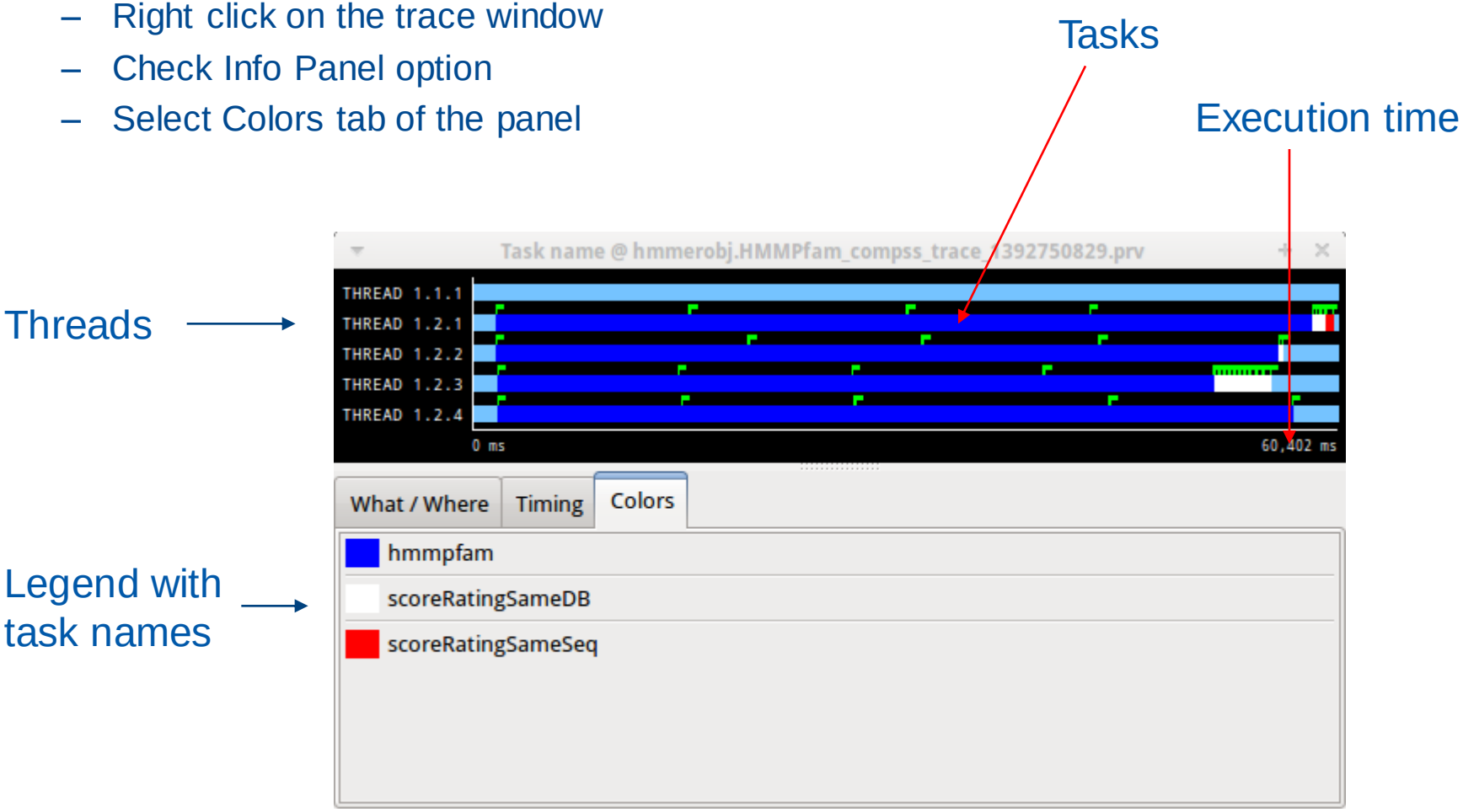


- View Event Flags
  - Right click on the trace window
  - View / Event Flags



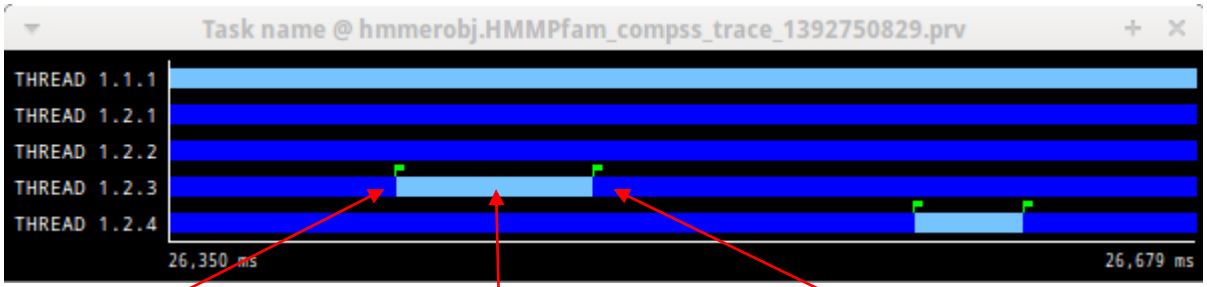
HMMER Hands On: Tracing

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



HMMER Hands On: Tracing

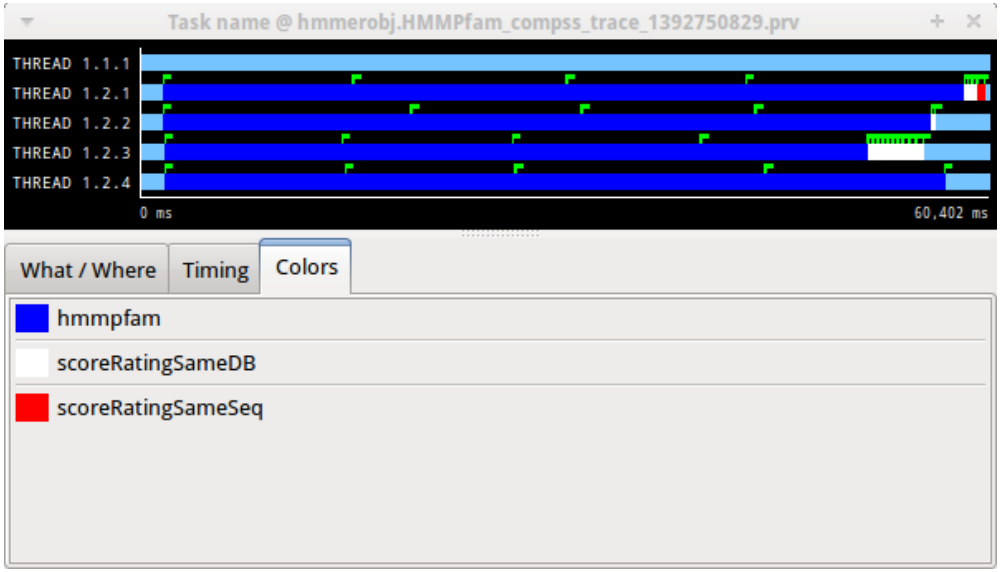
- Zoom to see details
  - Select a region in the trace window to see in detail
  - And repeat the process til the needed zoom level
  - The undo zoom option is in the right click panel



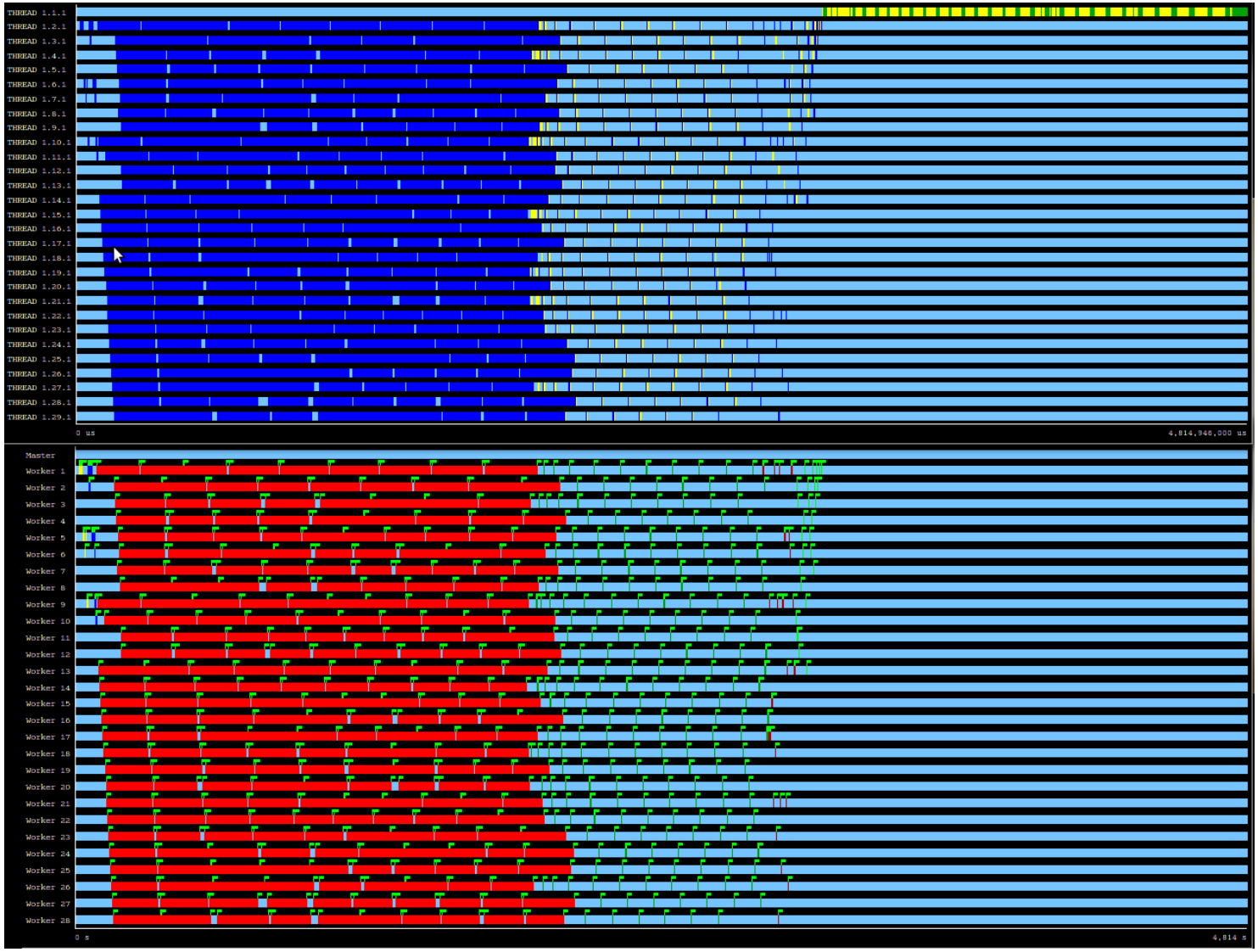
Previous task ends Processor is idle New task starts

HMMER Hands On: Tracing

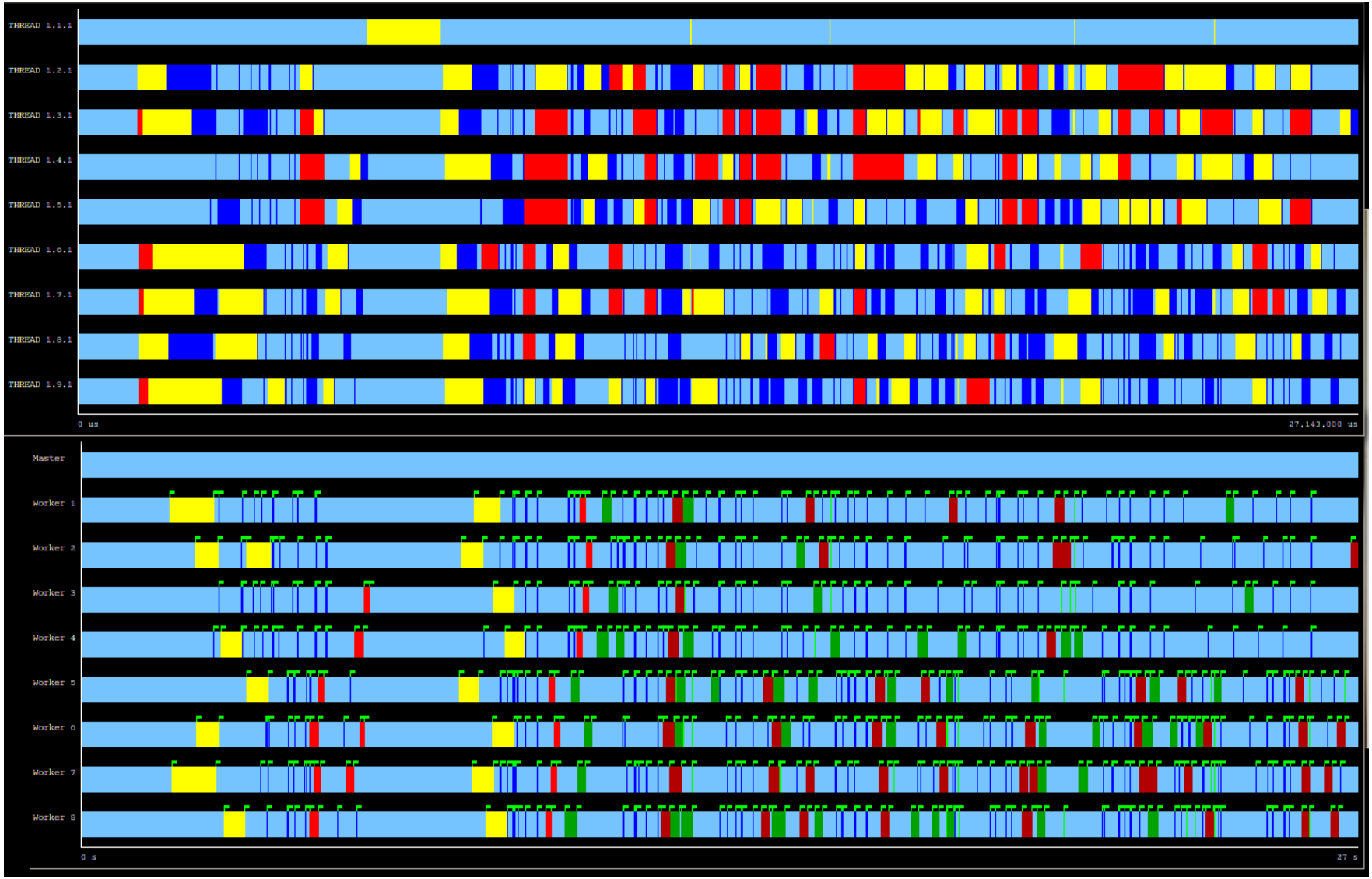
- Summarizing:
  - Lines in the trace:
  - One line for the master
  - N lines for the workers
- Meaning of the colours:
  - Light blue: idle
  - Other colors: task running
    - see the color legend
- Flags (events):
  - Start / end of task



Tracing: Other Examples



Tracing: Other Examples



Python Hands-on

Neuroscience Data Processing @ Parallel Python

Main program

```
#secret
scrt='mysecret'

# tuple of all parallel python servers to connect with
ppservers = ('comp1.my-network', 'comp2.my-network', 'comp3.my-network')

if len(sys.argv) > 1:
    ncpus = int(sys.argv[1])
    #creates jobserver with ncpus workers
    job_server = pp.Server(ncpus, ppservers=ppservers,  loglevel=0, secret=scrt)
else:
    #creates jobserver with automatically detected number of workers
    job_server = pp.Server(ppservers=ppservers,  loglevel=0, secret=scrt)

#wait for servers to come up
time.sleep(5)

#calculate number  of nodes in total
nlocalworkers = job_server.get_ncpus()
activenodes = job_server.get_active_nodes()
workerids = activenodes.keys()
nworkers=sum( [activenodes[workerids[i]] for i in range(len(workerids))] ) + nlocalworkers
num_ccs = (num_neurons**2 - num_neurons)/2

#calculate number  of pairs each worker should process
step = ceil(float(num_ccs)/nworkers)
start_idx = 0
end_idx = 0
starts = zeros((nworkers+1,))
starts[-1] = num_ccs

seed = 2398645
delta = 1782324
jobs = []
```

Function definition

```
def cc_surrogate_range(start_idx, end_idx, seed, num_neurons, num_surrs, num_bins, maxlag):
  …
```

Explicit resources declaration

Main program (cont)

```
for worker in range(nworkers):
    start_idx = end_idx
    end_idx = int(min((worker+1)*step,num_ccs))
    starts[worker] = start_idx
    params = start_idx, end_idx, seed, num_neurons, num_surrs, num_bins, maxlag
    depfuncs = ()
    depmodules = "numpy","pickle",
    jobs.append(job_server.submit(cc_surrogate_range,params,depfuncs,depmodules))
    seed = seed + delta

cc_original = zeros((num_ccs,2*maxlag+1))
cc_surrs = zeros((num_ccs,2*maxlag+1,2))
for worker in arange(nworkers):
    start = starts[worker]
    end = starts[worker + 1]
    result=jobs[worker]()
    cc_original[start:end,:] = result[0]
    cc_surrs[start:end,:,:] = result[1]

f = open('./result_cc_originals.dat','w')
pickle.dump(cc_original,f)
f.close()
f = open('./result_cc_surrogates_conf.dat','w')
pickle.dump(cc_surrs,f)
f.close()
```

Explicit fork join

Data back

Dump

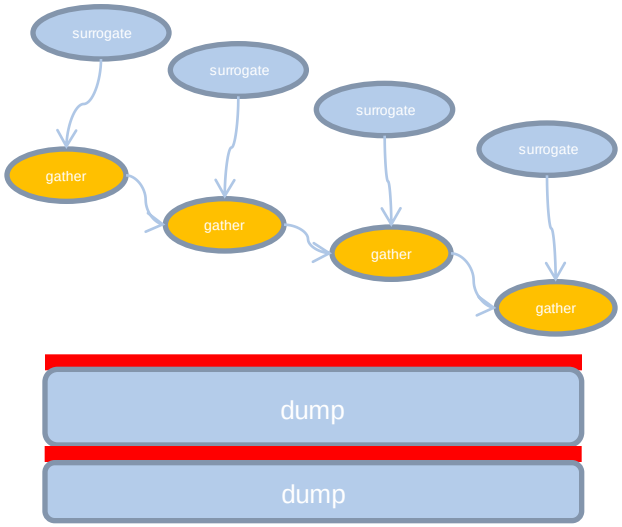
Neuroscience Data Processing @ PyCOMPSs

Main program

```
Import sys
from pycompss.api.api import compss_wait_on

num_frags = int(sys.argv[1])

#calculate number of pairs per fragment
num_ccs = (num_neurons**2 - num_neurons)/2
step = ceil(float(num_ccs)/num_frags)
start_idx = 0
end_idx = 0

seed = 2398645
delta = 1782324
```

Tasks definition

```
@task(...)
def gather(result, cc_original, cc_surrs, start, end):
    cc_original[start:end,:] = result[0]
    cc_surrs[start:end,:,:] = result[1]

@task(...)
def cc_surrogate_range(start_idx, end_idx, seed, num_neurons, num_surrs, num_bins, maxlag):
    …
```



Main program (cont)

```
cc_original = zeros((num_ccs,2*maxlag+1))
cc_surrs = zeros((num_ccs,2*maxlag+1,2))
for frag in range(num_frags):
    start_idx = end_idx
    end_idx = int(min((frag+1)*step,num_ccs))
    result = cc_surrogate_range(start_idx, end_idx, seed, num_neurons, num_surrs, num_bins, maxlag)
    gather(result, cc_original, cc_surrs, start_idx, end_idx)
    seed = seed + delta

f = open('./result_cc_originals.dat','w')
cc_original = compss_wait_on(cc_original)
pickle.dump(cc_original,f)
f.close()

f = open('./result_cc_surrogates_conf.dat','w')
# sync!
pickle.dump(cc_surrs,f)
f.close()
```

PyCOMPSs Hands On: Exercise

- Complete task definition
- Add missing object synchronization

PyCOMPSs Hands-on: Neurons Execution

- Execution with CLI

```
runcompssext --lang=python \
--app=$HOME/python_workspace/neurons-python/neurons/ns-data-proc_compss_objects.py \
--classpath=$HOME/python_workspace/neurons-python \
--cline_args="16 $HOME/python_workspace/neurons-python/neurons/spikes.dat"
```

/home/user/launch_neurons_py.sh

- Execution from Eclipse
  - Run-> External Tools

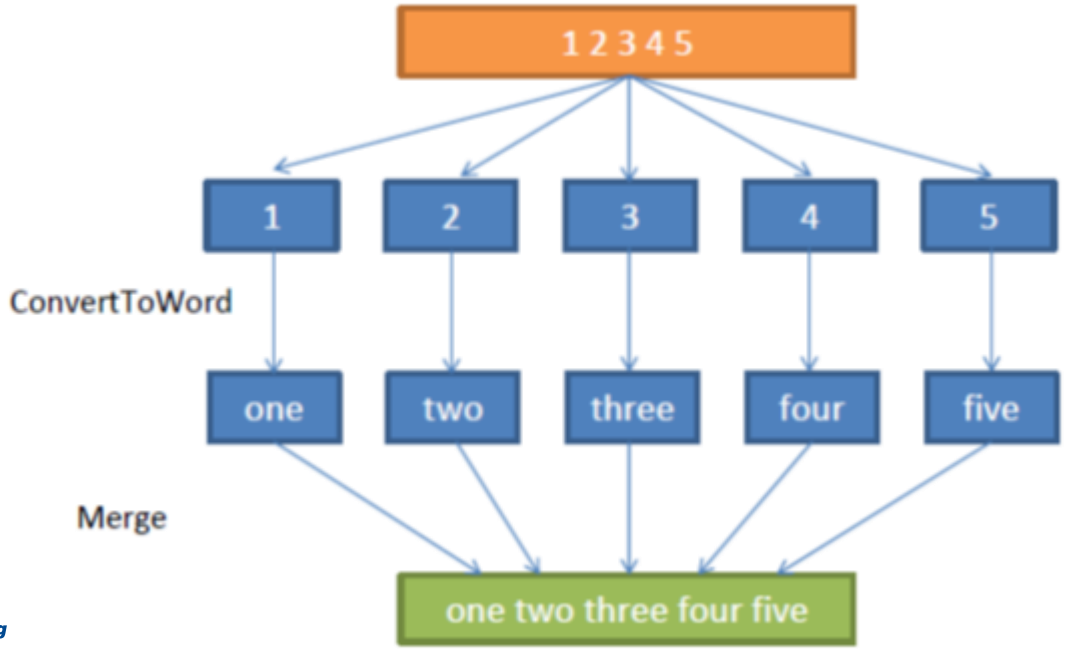
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Integrated Development Environment
HANDS-ON

IDE Hands On: Convert number to words application

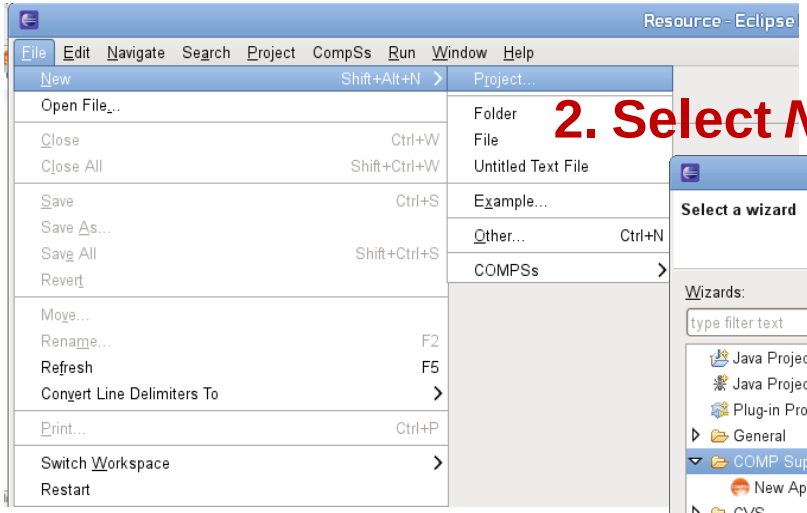
- Input application args numbers to convert e.g: 1 2 3 4 5
- Expected result: one two three four five
- Pseudo-code

```
for each number
        word <- convertToWord(number)
        result <- Concatenate( result, word)
```

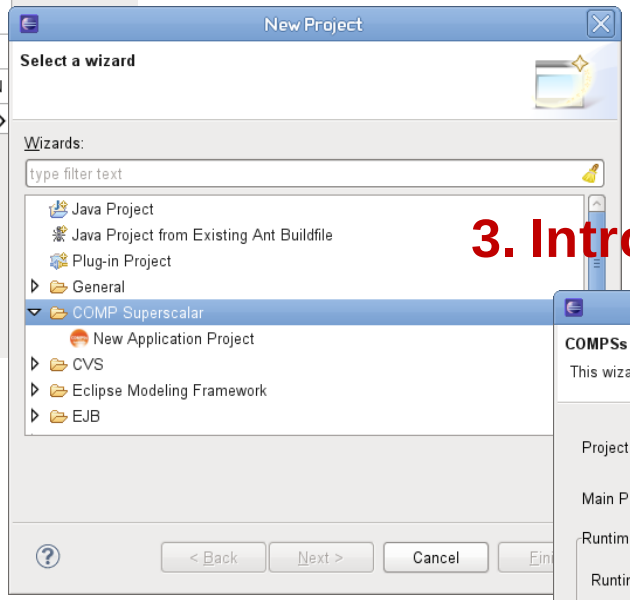


IDE Hands On – Create a COMPSs Project

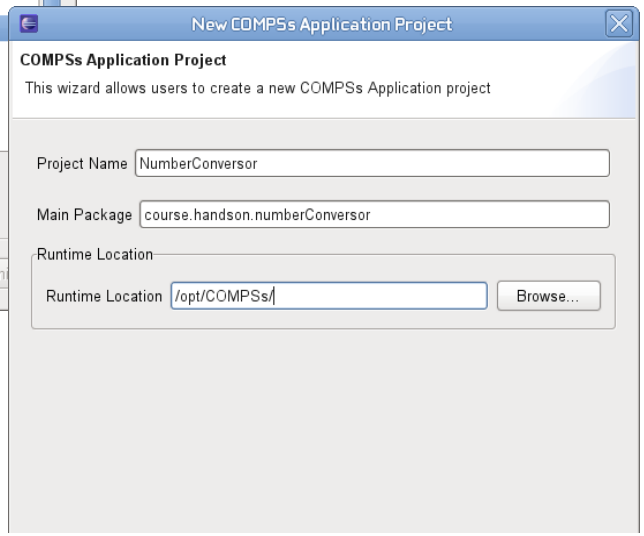
1. Menu *File->New -> Project…*



2. Select *New Application Project*



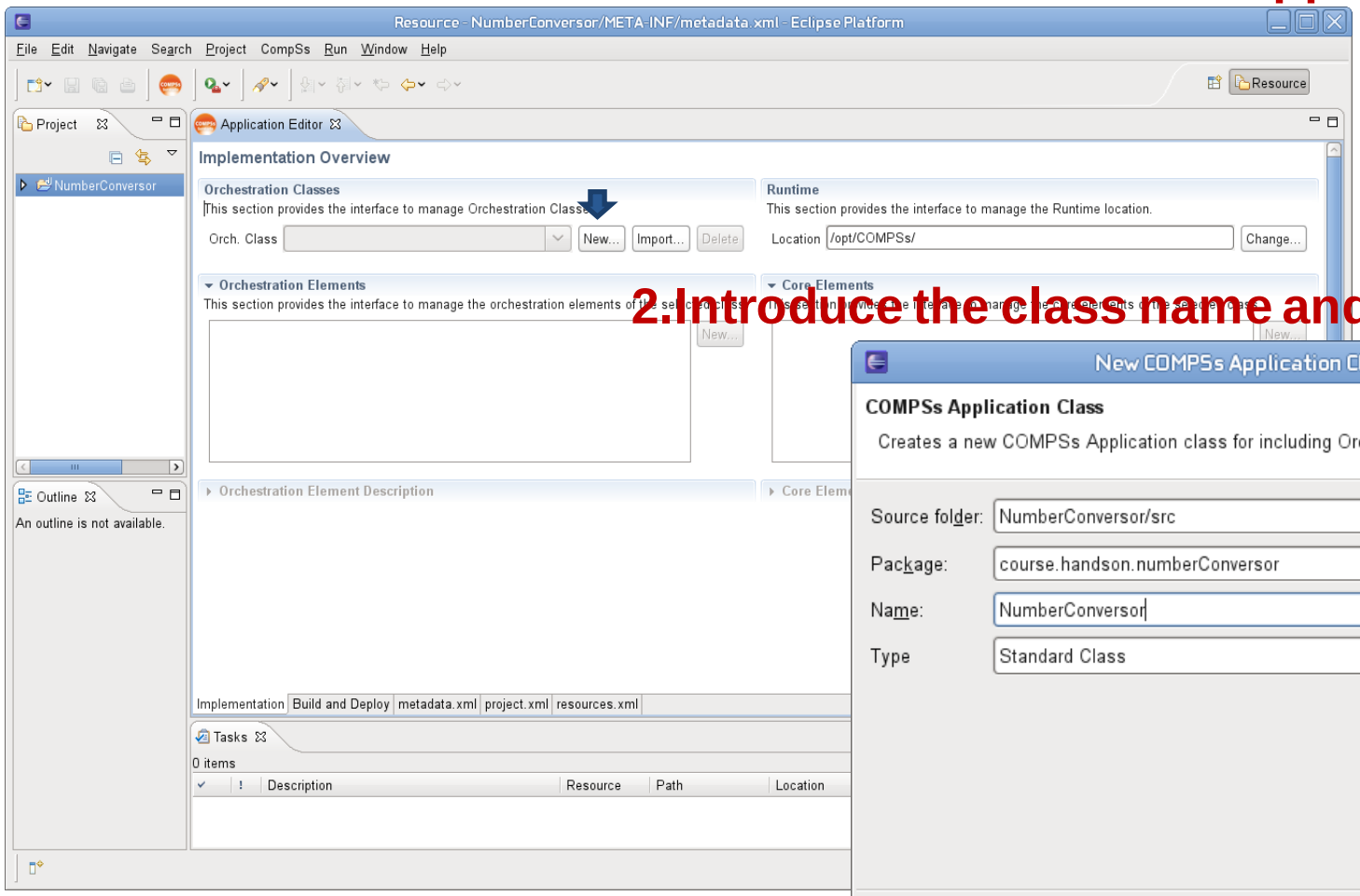
3. Introduce Project Details



(Also available *CompSs –>Implementation->Create Application Project*)

IDE Hands On – Create an Orchestration Class

1. Click *New…* in Orchestration Classes section of Application Editor



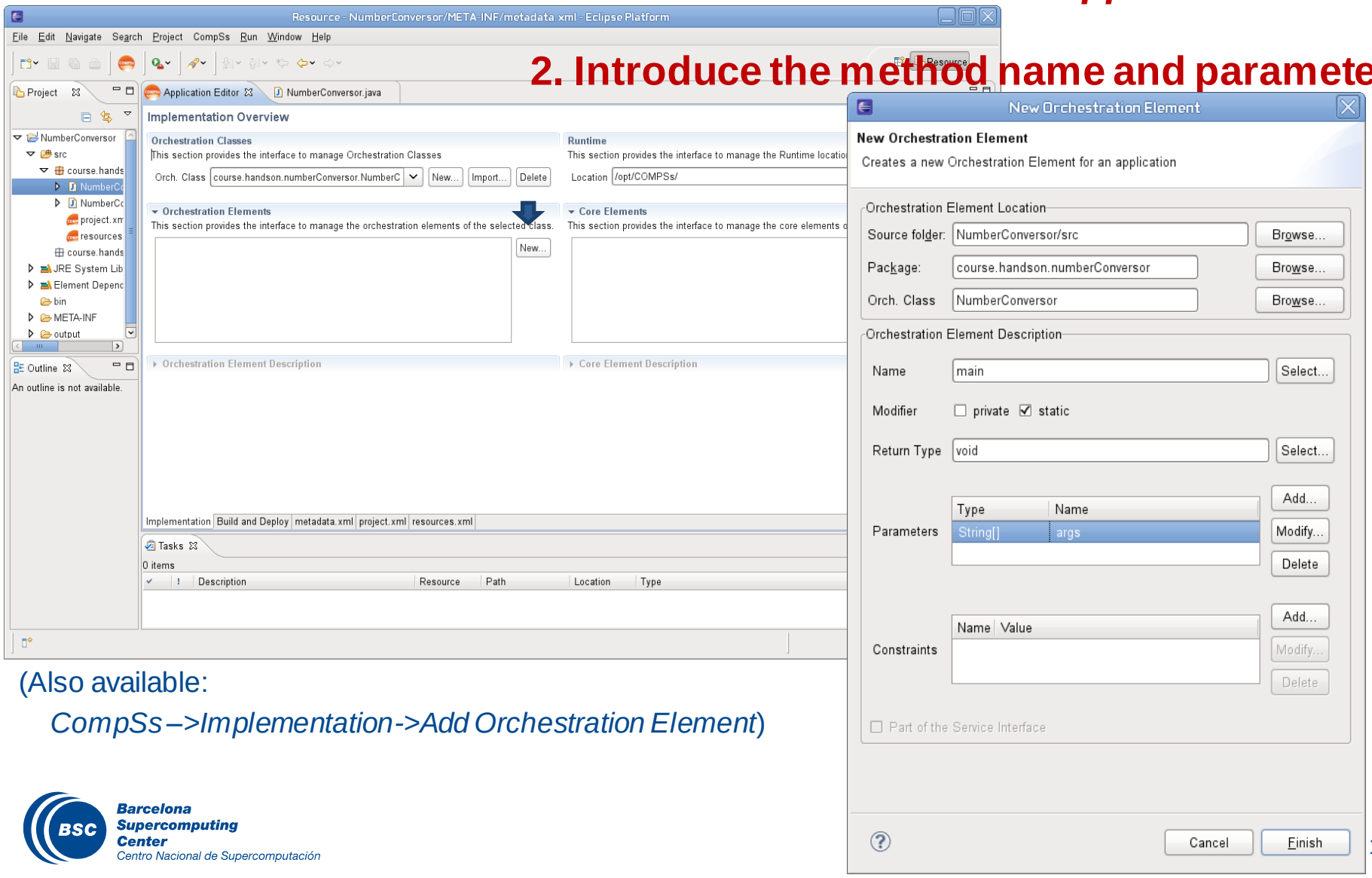
2.Introduce the class name and type (Standard)

(Also available *CompSs –>Implementation->Create Orchestration Class*)

IDE Hands On – Create an Orchestration Element

1. Click *New…* in *Orchestration Elements* sec. of the *Application Editor*

2. Introduce the method name and parameters



(Also available:

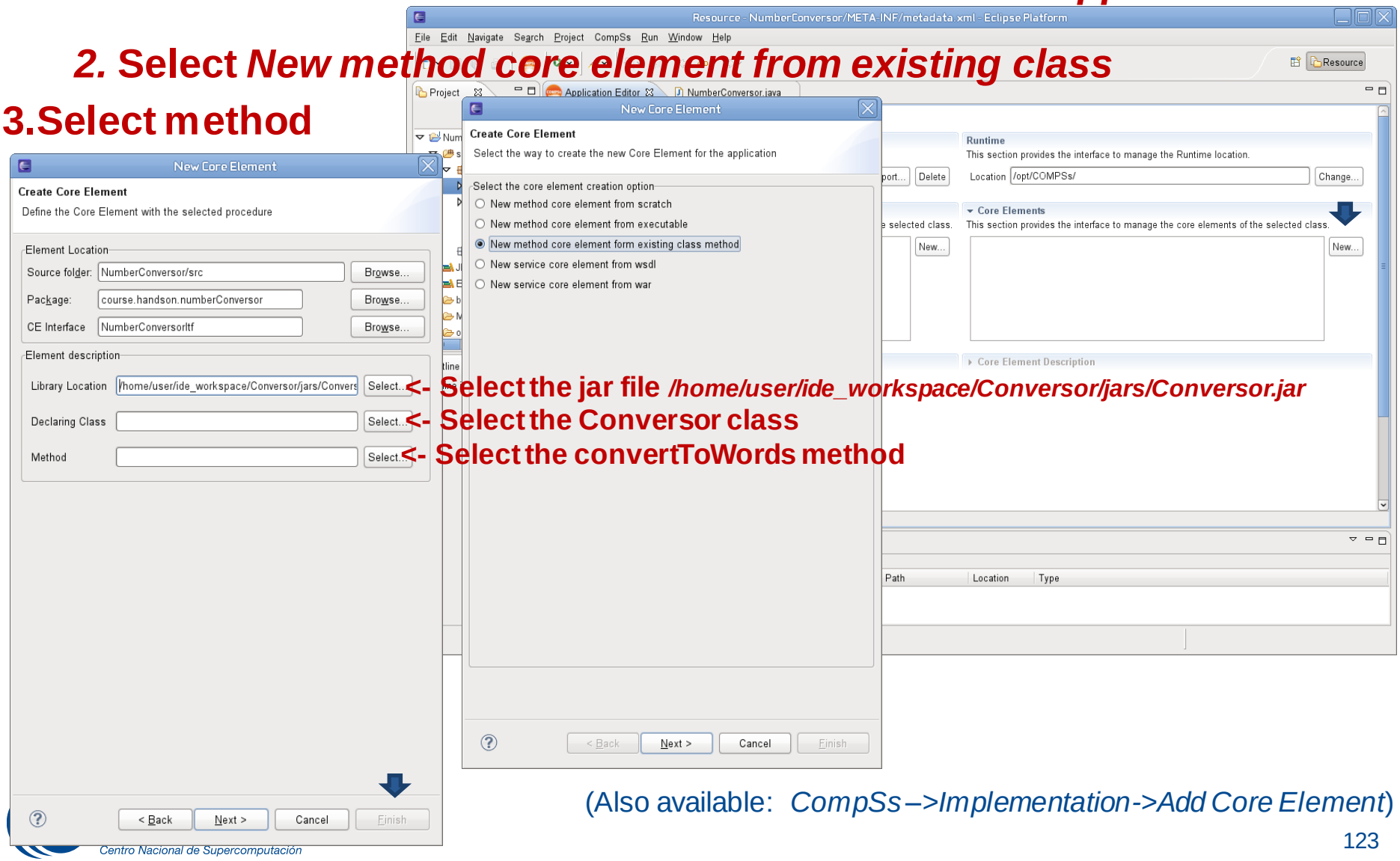
CompSs –>Implementation->Add Orchestration Element)

IDE Hands On – Add an Core Element from JAR

1. Click *New…* in *Core Elements* section of the *Application Editor*

2. Select *New method core element from existing class*

3. Select method



<- Select the jar file */home/user/ide_workspace/Conversor/jars/Conversor.jar*

<- Select the Conversor class

<- Select the convertToWords method

(Also available: *CompSs –>Implementation->Add Core Element*)

IDE Hands On – Add an Core Element from scratch

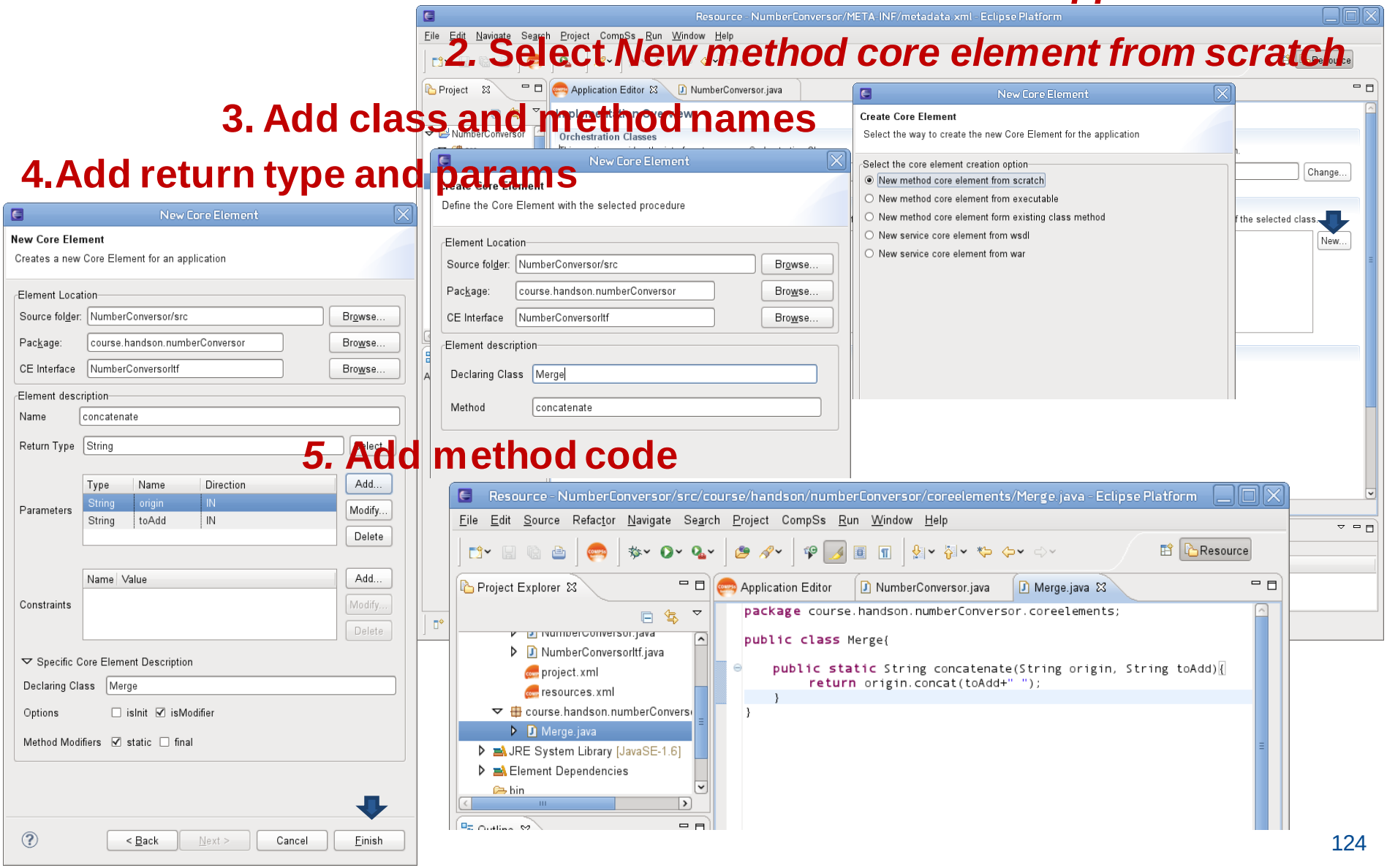
1. Click *New…* in *Core Elements* section of the *Application Editor*

2. Select *New method core element from scratch*

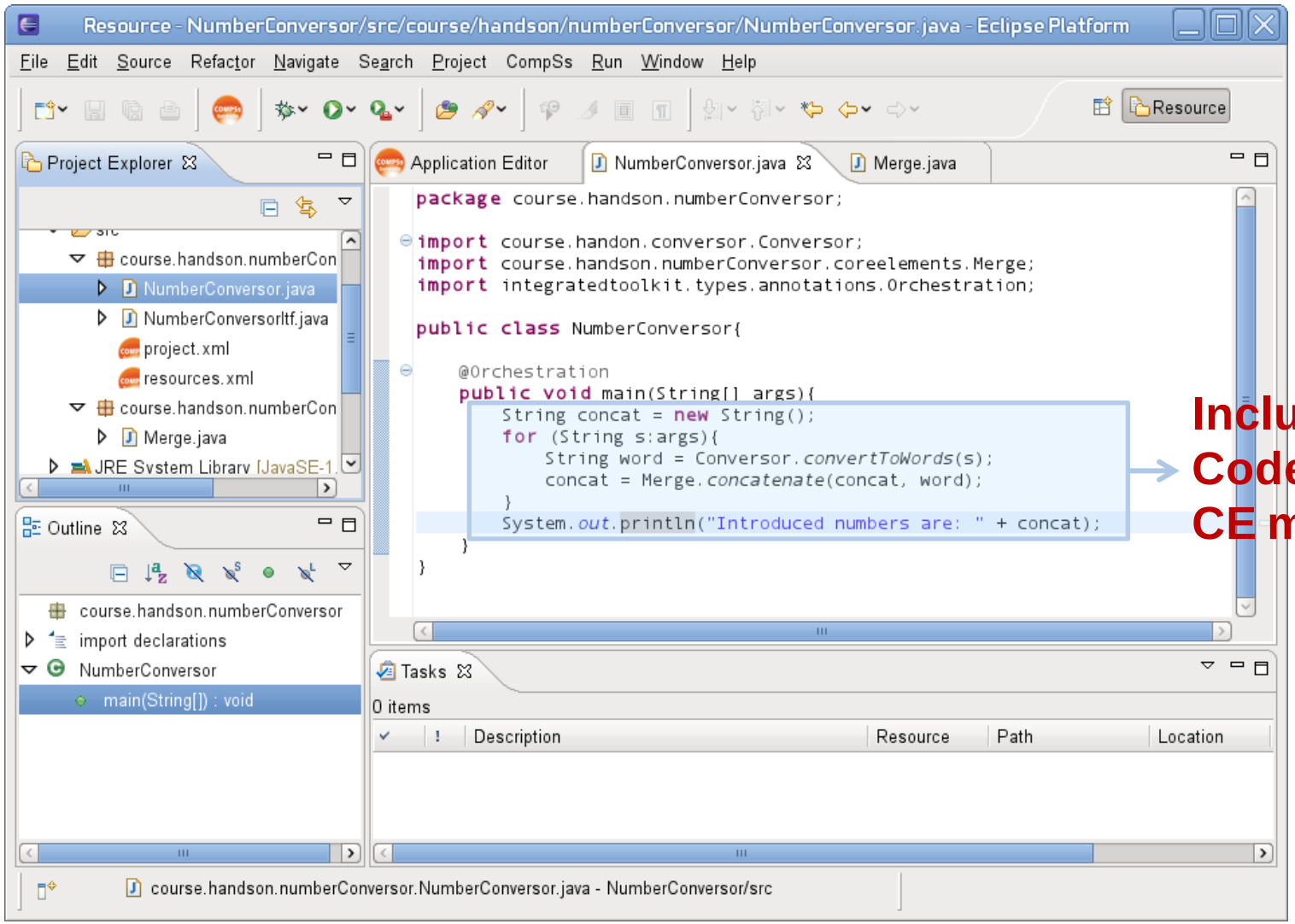
3. Add class and method names

4. Add return type and params

5. Add method code

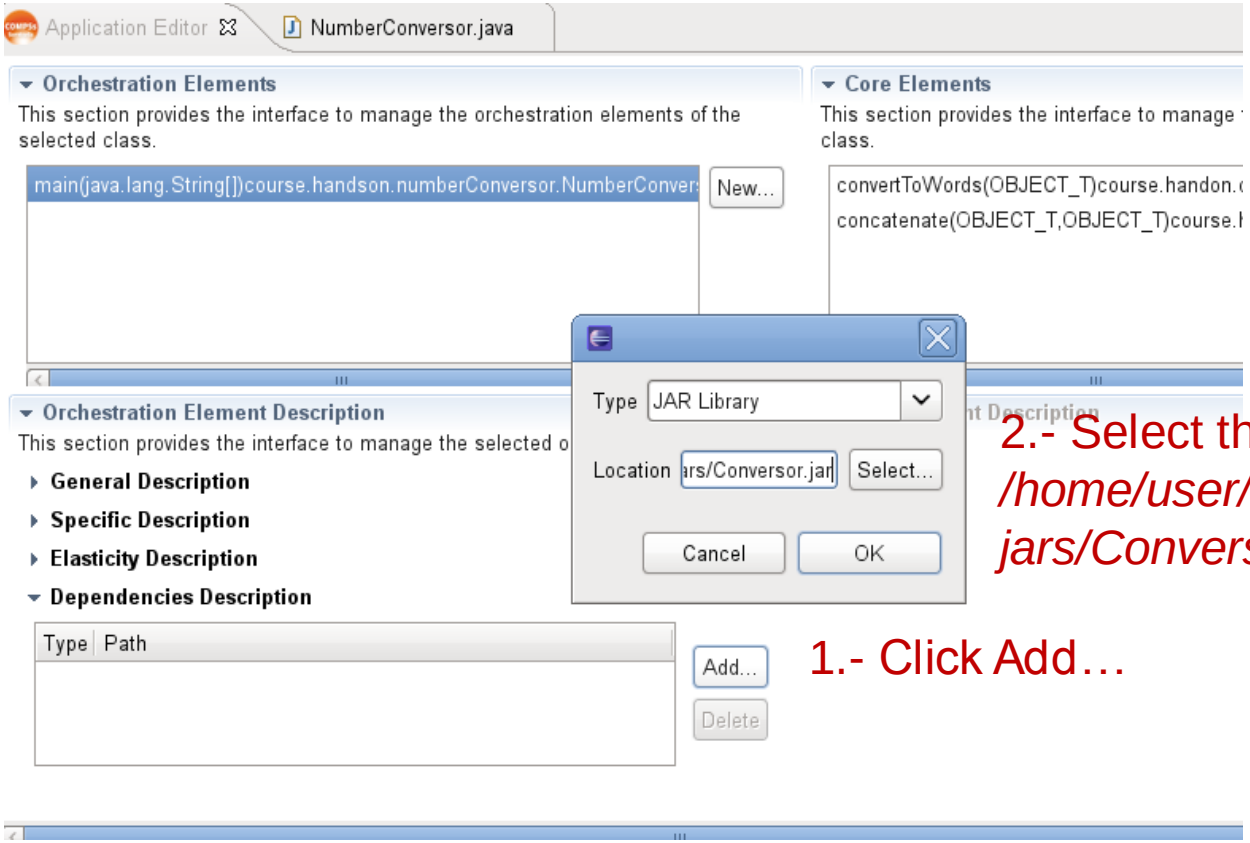


IDE Hands On – Introduce the OE code



Include the OE Code to call the CE methods

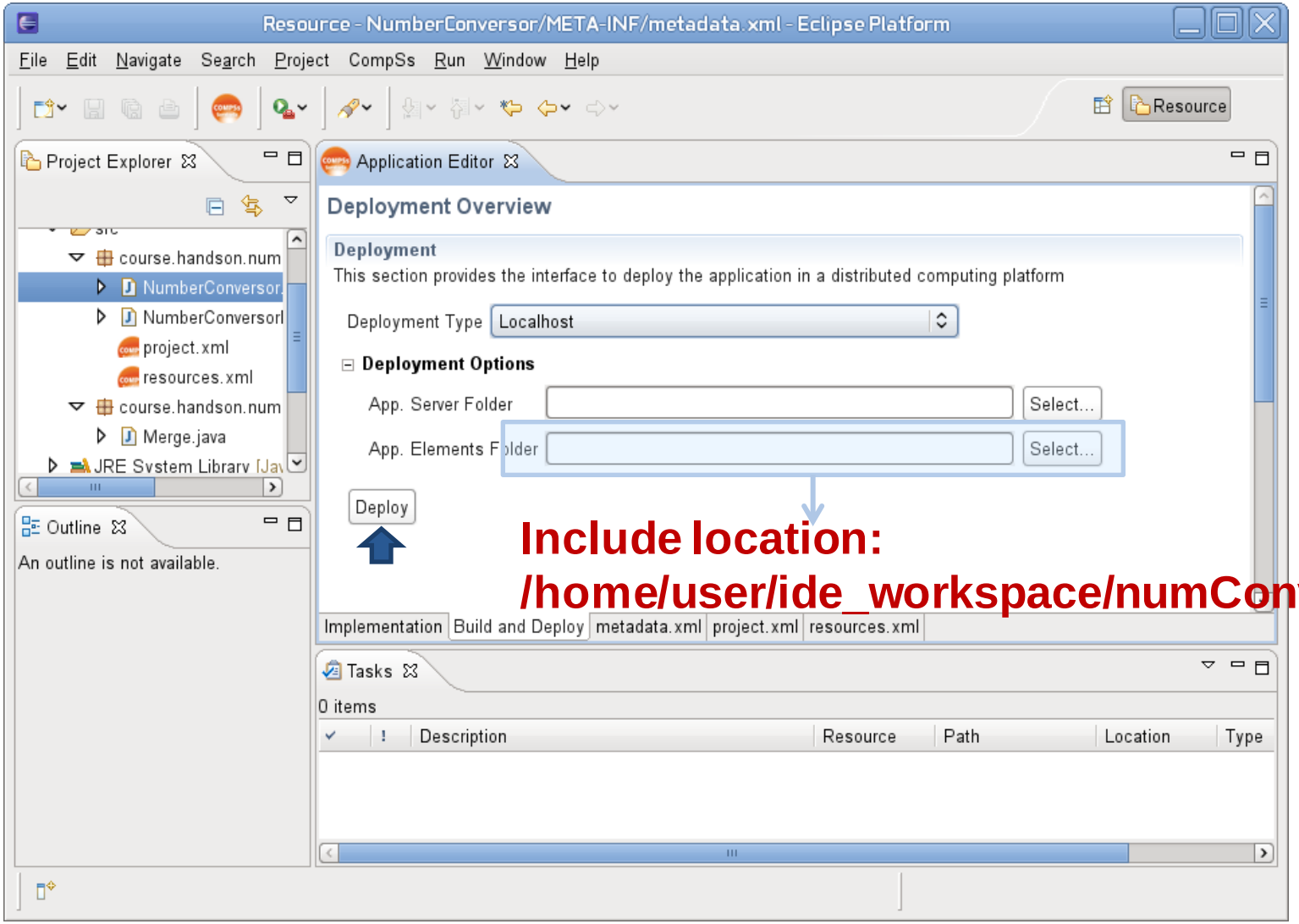
IDE Hands On – Add conversor dependency to OE



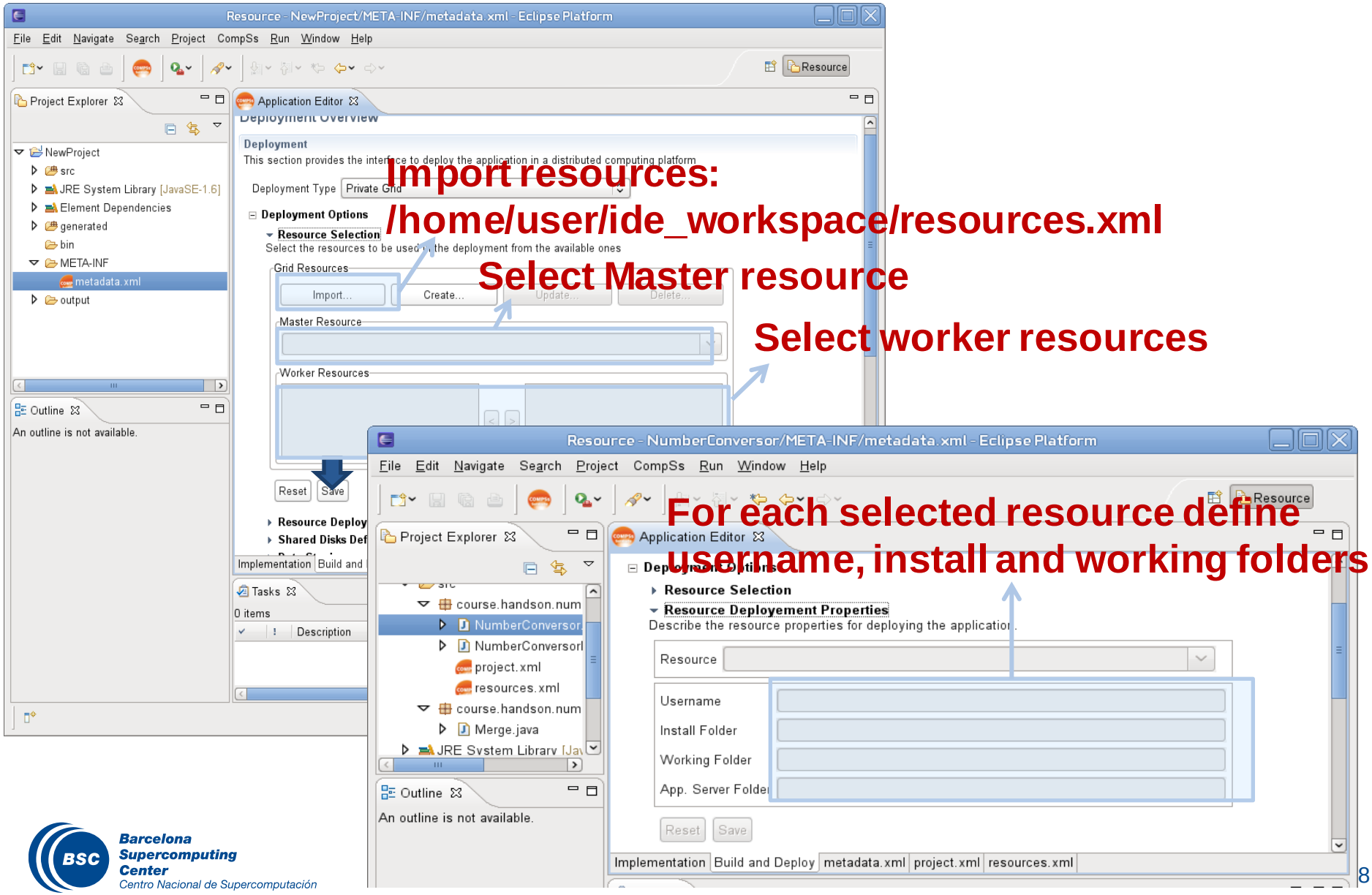
2.- Select the jar library: */home/user/ide_workspace/Conversor/jars/Conversor.jar*

1.- Click Add…

IDE Hands On – Deploy Locally



IDE Hands On – Deploy Grid



Final Notes

- 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
  - Web services

Final Notes

- Project page:
  - http://www.bsc.es/compss
- Direct downloads page:
  - http://www.bsc.es/computer-sciences/grid-computing/comp-superscalar/download
    - 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

www.bsc.es

Barcelona Supercomputing Center

Centro Nacional de Supercomputación

Thank you!

For further information please contact

support-compss@bsc.es