

[www.bsc.es](http://www.bsc.es)



**Barcelona  
Supercomputing  
Center**

*Centro Nacional de Supercomputación*

# COMPSs Tutorial

February 20th 2014, Barcelona

Rosa M. Badia, Carlos Diaz, Jorge Ejarque  
Daniele Lezzi, Francesc Lordan  
Roger Rafanell, Raül Sirvent  
Enric Tejedor

# Outline (Feb 20<sup>th</sup> 2014)

- ⌘ **Session 1 / 9am – 11am: Introduction to COMPSs**
- ⌘ **Roundtable: presentation and background of participants**
- ⌘ **Programming model**
  - **Overview**
  - **Steps**
  - **Properties**
- ⌘ **COMPSs runtime system**
  - **Overview**
  - **Features**
- ⌘ **Coffee break – 11:00 – 11:30**
- ⌘ **Session 2 / 11:30am – 1pm: Application examples**
  - **Sample codes & Demos**
    - **Matmul**
  - **Graphical interface (IDE)**
    - **Gene Detection**

# Outline (Feb 20<sup>th</sup> 2014)

**⌘ Lunch Break 1pm to 2pm**

**⌘ Session 3 / 2 pm- 3:30 pm: Hands-on I**

- Virtual Machine Setup
- BLAST overview
- Code modification
  - All-to-one and tree-reduction
- Compilation and Execution

**⌘ Coffee break: 3:30 – 4:00**

**⌘ Session 4 / 4 pm- 6 pm: Hands-on II**

- HMMER overview & code modification
- Configuration, monitoring, debugging
- Overview of tracing, trace analysis
- IDE Hands-on
- Final notes

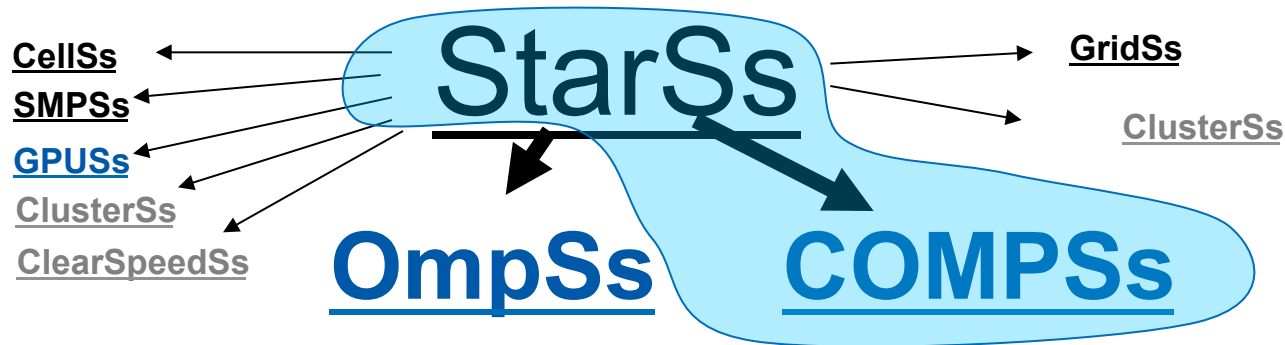


**Barcelona  
Supercomputing  
Center**

*Centro Nacional de Supercomputación*

# COMPSs Programming Model

# The StarSs programming model



## « StarSs

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

## « OpenSource

- <http://compps.bsc.es>

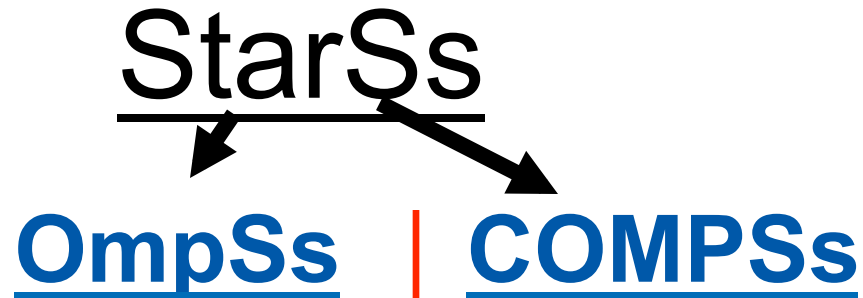
## ▶ Programmability/Portability

- ▶ “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

# The StarSs programming model: granularities



## ⌘ 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

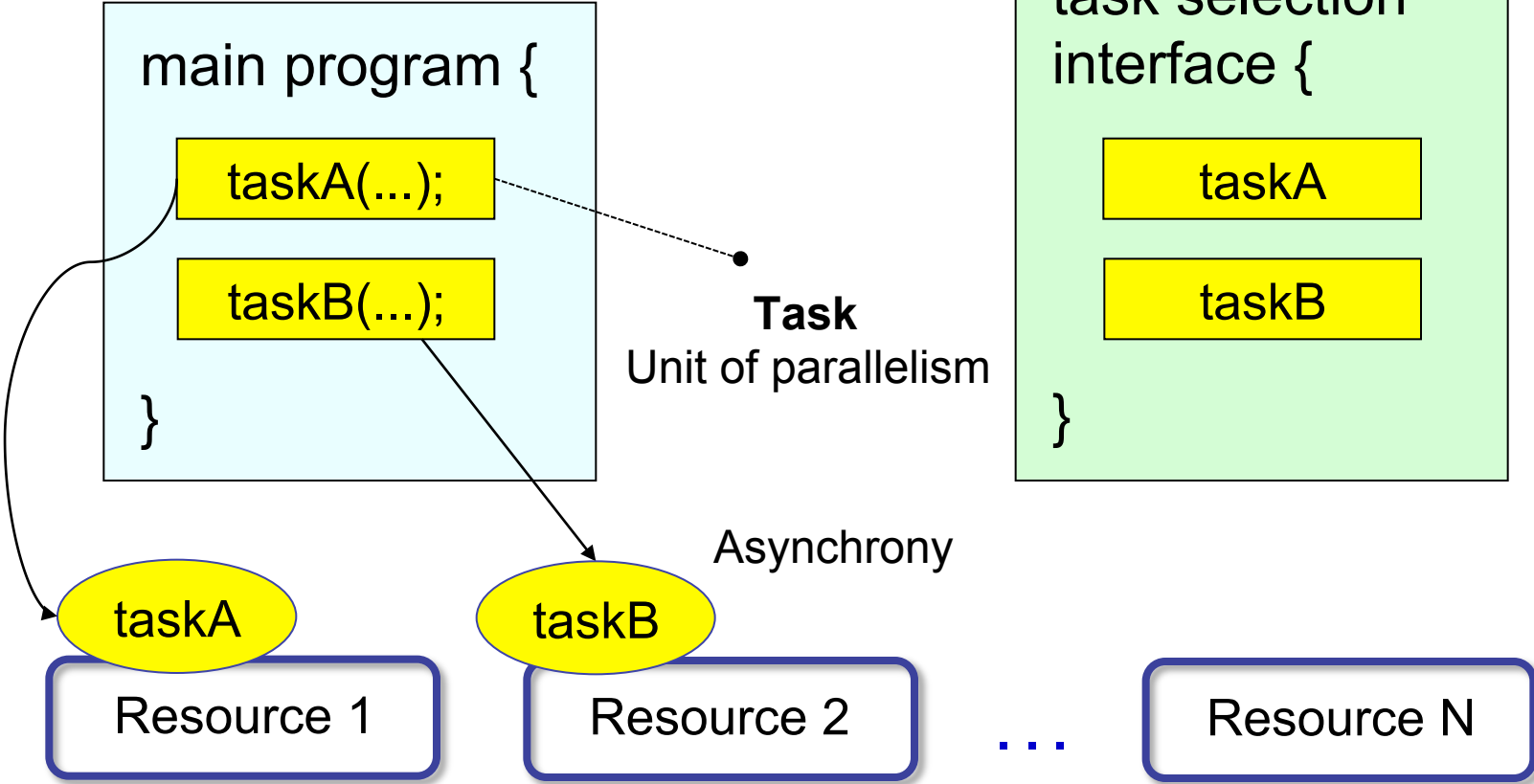
# Overview: Objectives

- « 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

## 1. Identify tasks

## 2. Select tasks



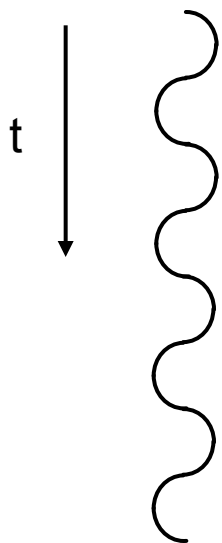


# Programming Model: Properties (I)

## Based on pure-Java fully-sequential programming

- No APIs, no threading, no messaging
- No parallel constructs, no pragmas
- Sequential consistency

main thread



```
Main Program {  
    taskA(data1);  
  
    for (int i=0; i< N; i++)  
        taskB(data1, data2);  
  
    if (condition)  
        process(data2);  
}
```

taskA

taskB

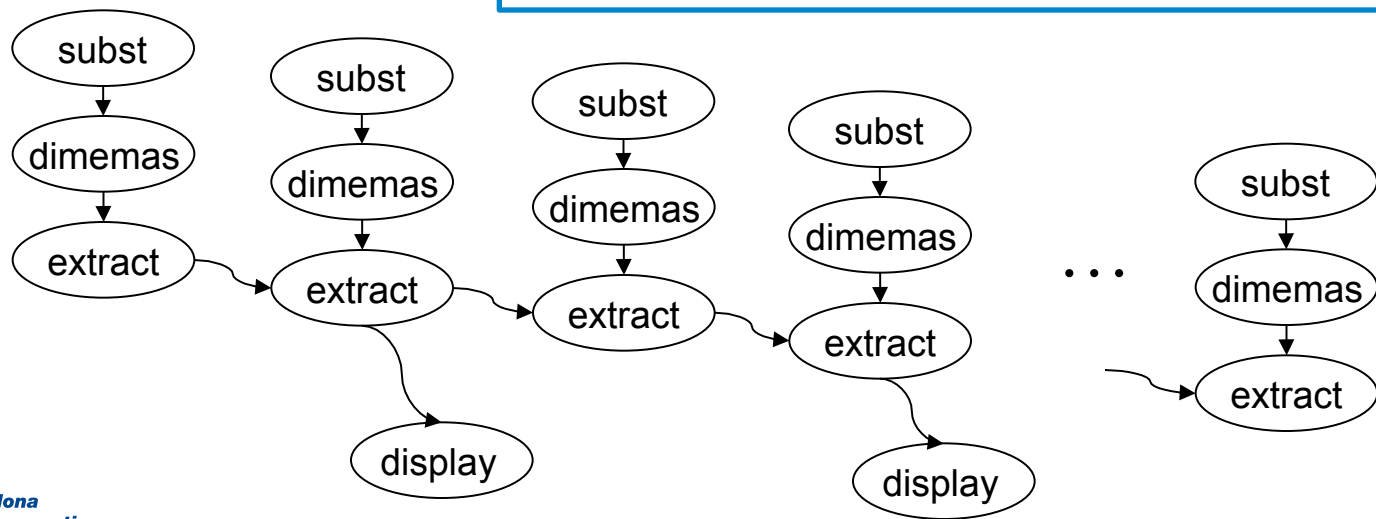
synch

# 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

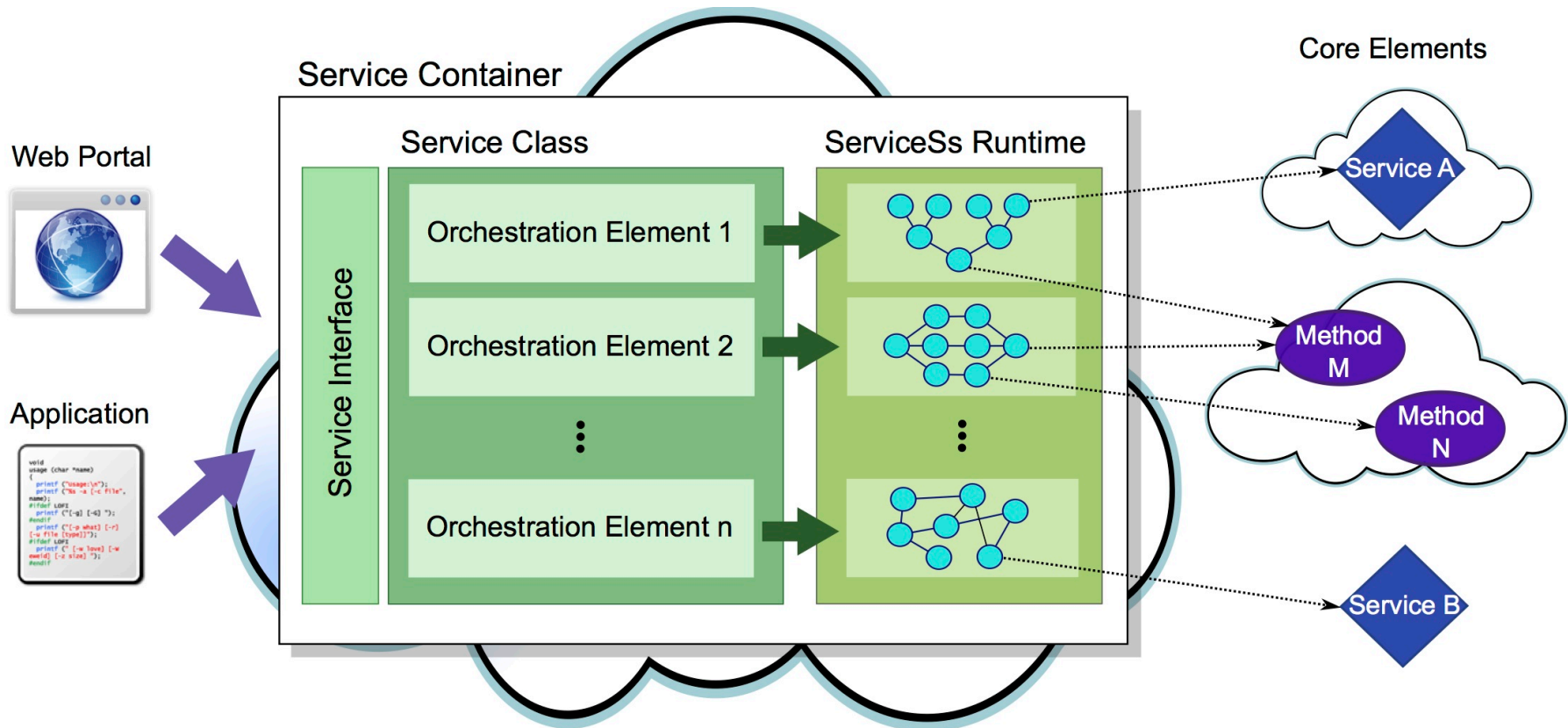


Cluster



Cloud

# Introduction to COMPSs



# 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  
    );  
}
```

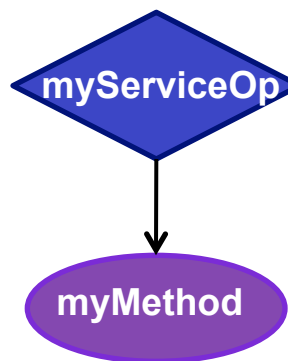
# Programming Model: Regular Main program

```
public class App {  
  
    public static void main(String[] args) {  
        Query query = new Query(...);  
        Reply reply = myServiceOp(query);  
  
        myMethod(reply);  
  
        reply.printToLog();  
    }  
}
```

Service task call

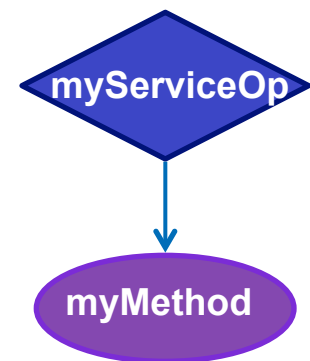
Method task call

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

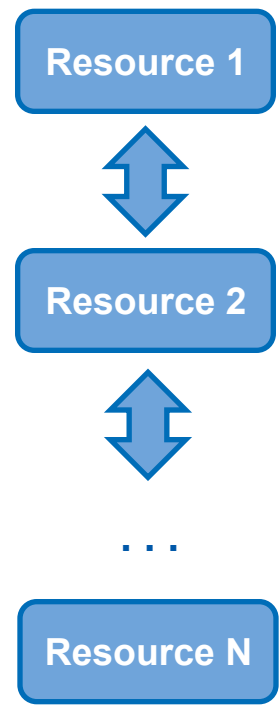
## Sequential Code

```
...  
for (i=0; i<N; i++){  
  T1 (data1, data2);  
  T2 (data4, data5);  
  T3 (data2, data5, data6);  
  T4 (data7, data8);  
  T5 (data6, data8, data9);  
}  
...
```

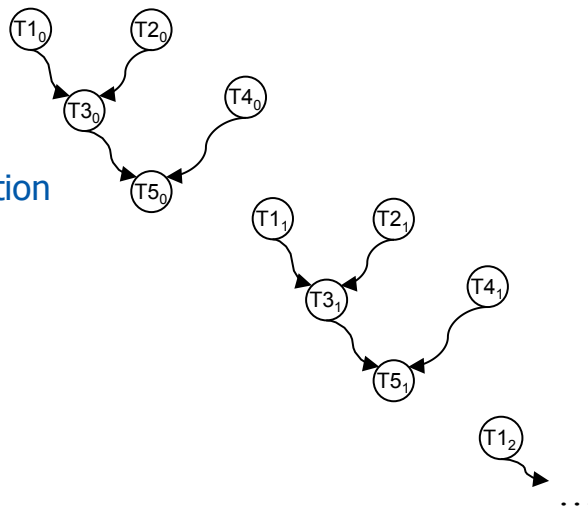
(a) Task selection +  
parameters direction  
(input, output, inout)

(d) Task completion,  
synchronization

## Parallel Resources



(b) Task graph creation  
based on data  
dependencies



(c) Scheduling,  
data transfer,  
task execution



# Programming Model: Sample Application

```
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);  
    }  
}
```

**Main program**

**Subroutine**

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

# Programming Model: Sample App (Interface)

## Task selection interface

```
public interface SimpleItf {
```

```
    @Method(declaringClass = "SimpleImpl")
```

```
    void increment(
```

```
        @Parameter(type = FILE, direction = INOUT)
```

```
        String counterFile
```

```
    );
```

```
}
```

**Implementation**



**Parameter  
metadata**



# Programming Model: Final App Code

```
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);  
    }  
}
```

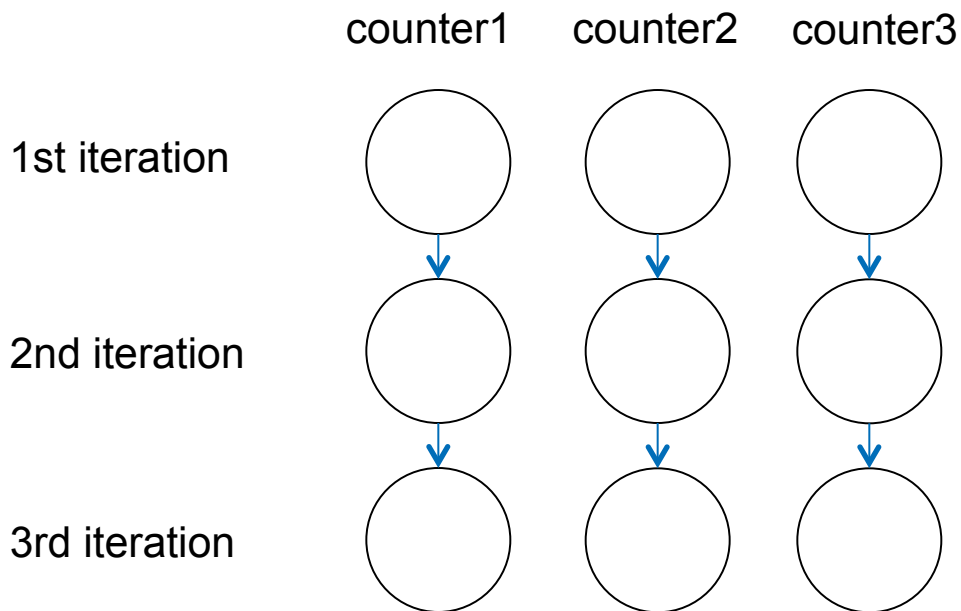
**Main program  
of the application  
NO CHANGES!**

# Programming Model: Task Graph

## Main loop

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

## Task graph



# 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



**Barcelona  
Supercomputing  
Center**

*Centro Nacional de Supercomputación*

# COMPSs Runtime System

# Runtime System

Application

Task Selection Interface

Runtime System



Grid



Cluster



Cloud



# Basic features

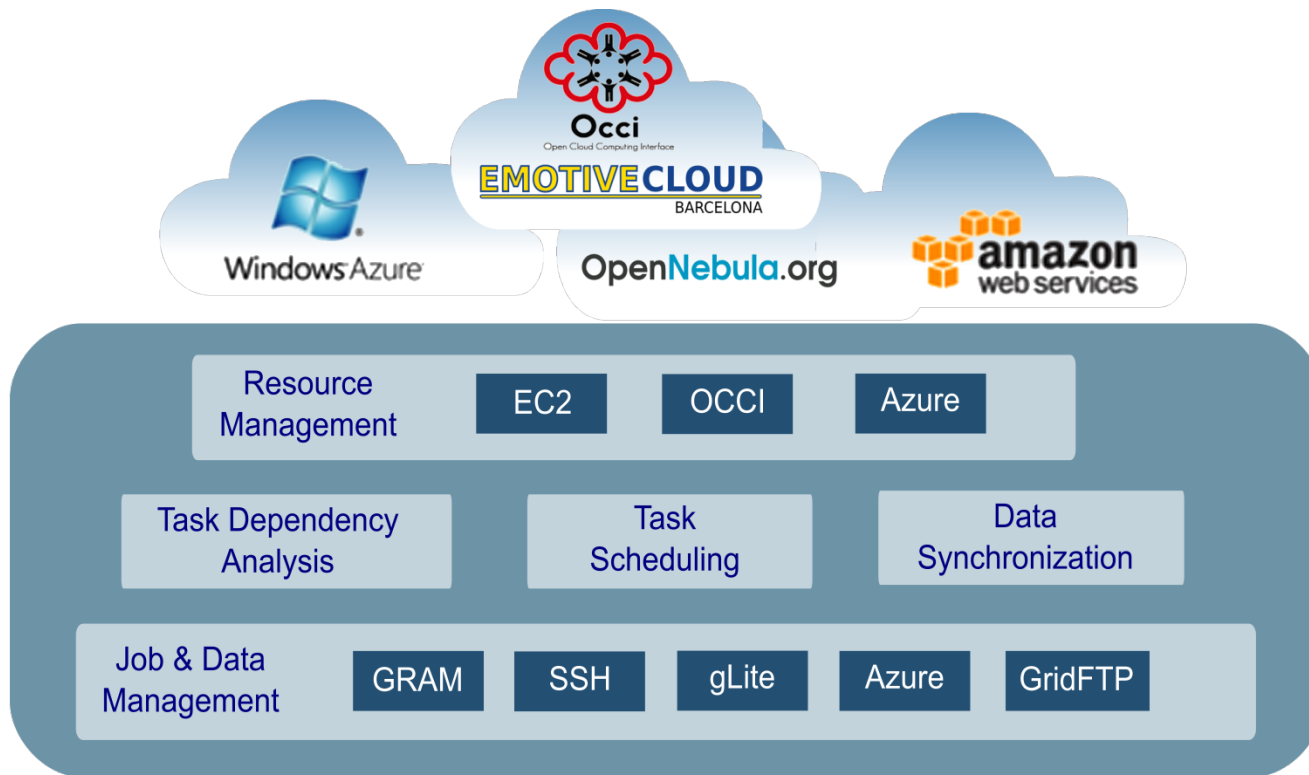
## « Supported Features:

- Data dependency analysis
- Data renaming
- Data transfer
- Task scheduling
- Resource management
- Results collection
- Fault tolerance
- Shared disks management

## « In Progress:

- Checkpointing
- Constraint matching
- Task nesting

# Interoperability



# 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>
        <CPUCount>1</CPUCount>
      </Processor>
      <OS>
        <OSType>Linux</OSType>
        <MaxProcessesPerUser>32</MaxProcessesPerUser>
      </OS>
      <StorageElement>
        <Size>30</Size>
      </StorageElement>
    </Capabilities>
  </Resource>
  ...
</ResourceList>
```

```
...
  <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/</InstallDir>
    <WorkingDir>/tmp/</WorkingDir>
    <User>user</User>
    <LimitOfTasks>1</LimitOfTasks>
  </Worker>

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

# Cloud Configuration: Resources Specification

## Resources.xml

```
<ResourceList>
  <CloudProvider name="BSCCloud">
    <Server>https://bscgrid20.bsc.es:8443/DRP</Server>
    <Connector>
      integratedtoolkit.connectors.emotivecloud.DRPSecureClientConnector
    </Connector>
    <ImageList>
      <Image name="debianbase"/>
    </ImageList>
    <InstanceTypes>
      <Resource Name="bsc.small">
        <Capabilities>
          <Processor>
            <CPUCount>1</CPUCount>
          </Processor>
          <StorageElement>
            <Size>0.5</Size>
          </StorageElement>
          <Memory>
            <PhysicalSize>1</PhysicalSize>
          </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>

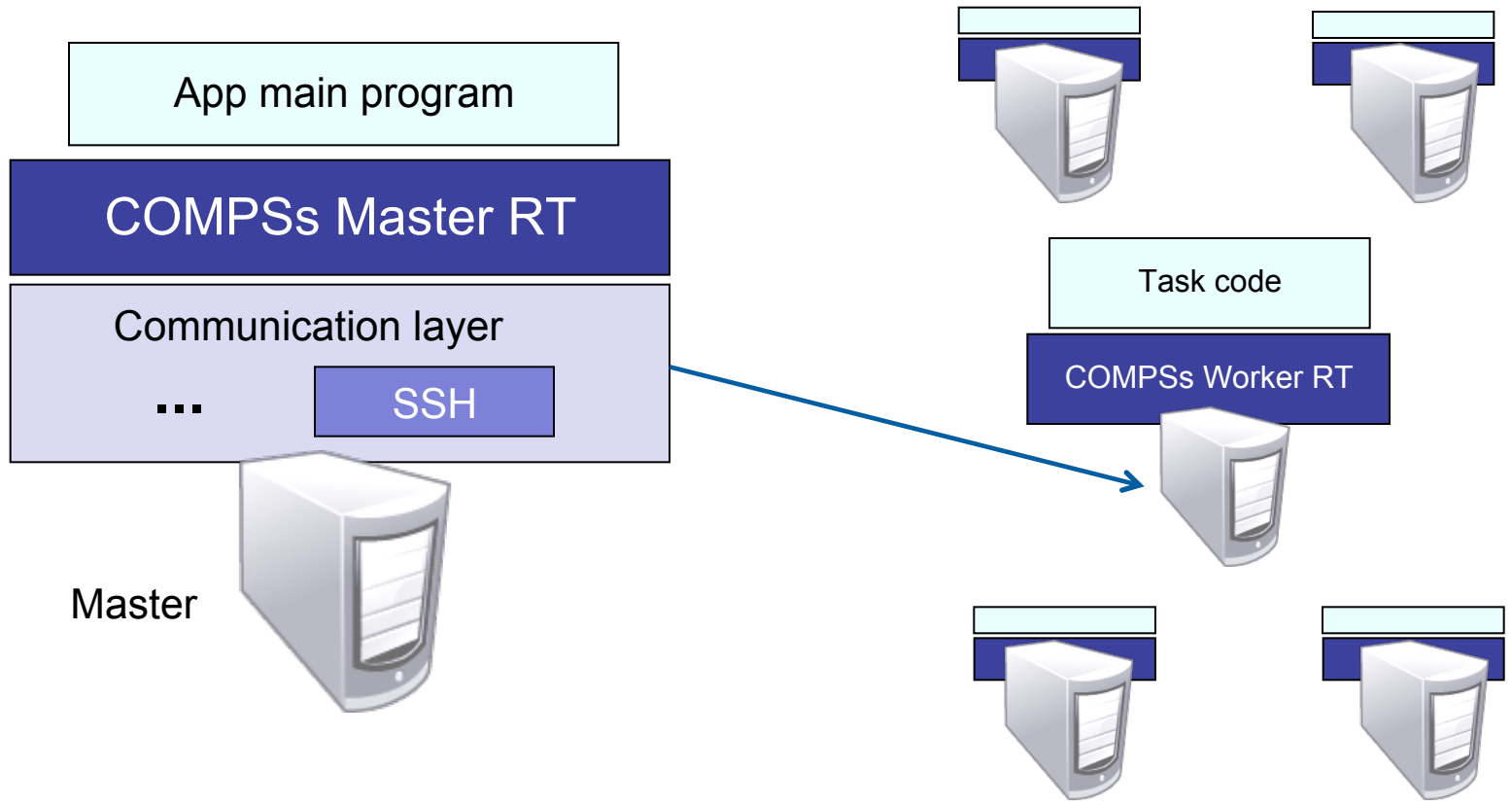
    <Provider name="BSCCloud">
      <LimitOfVMs>5</LimitOfVMs>
      <Property>
        <Name>Cert</Name>
        <Value>/home/.../cert.p12</Value>
      </Property>
      <Property>
        <Name>Owner</Name>
        <Value>userbsc</Value>
      </Property>
      <Property>
        <Name>JobNameTag</Name>
        <Value>Job</Value>
      </Property>
    ...
  </Cloud>
</Project>
```

```
...
<ImageList>
  <Image name="debianbase">
    <InstallDir>/opt/COMPSS/Runtime/scripts</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>
```

# COMPSs in a Cluster (interactive)

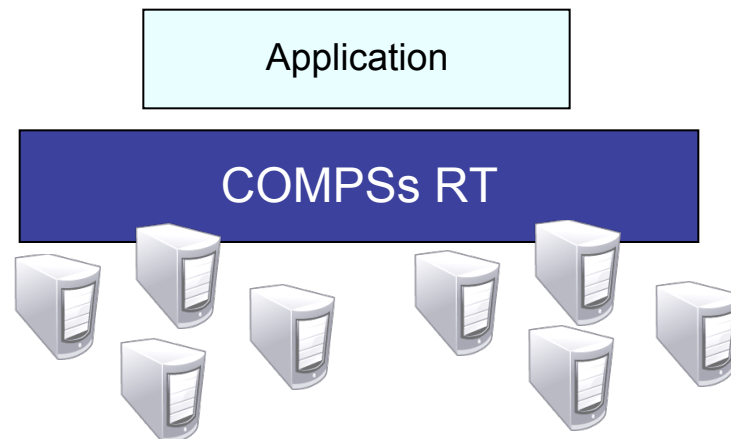
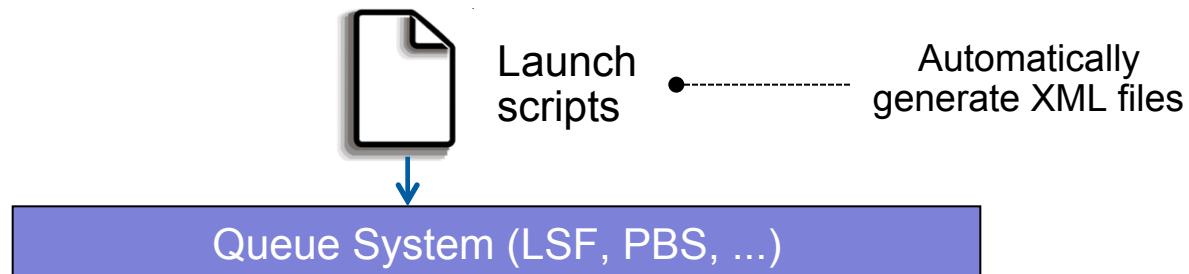
- Typical setup:
  - Master node: main program (+ master runtime)
  - Worker nodes: tasks (+ worker runtime)

Described by XML files  
Workers



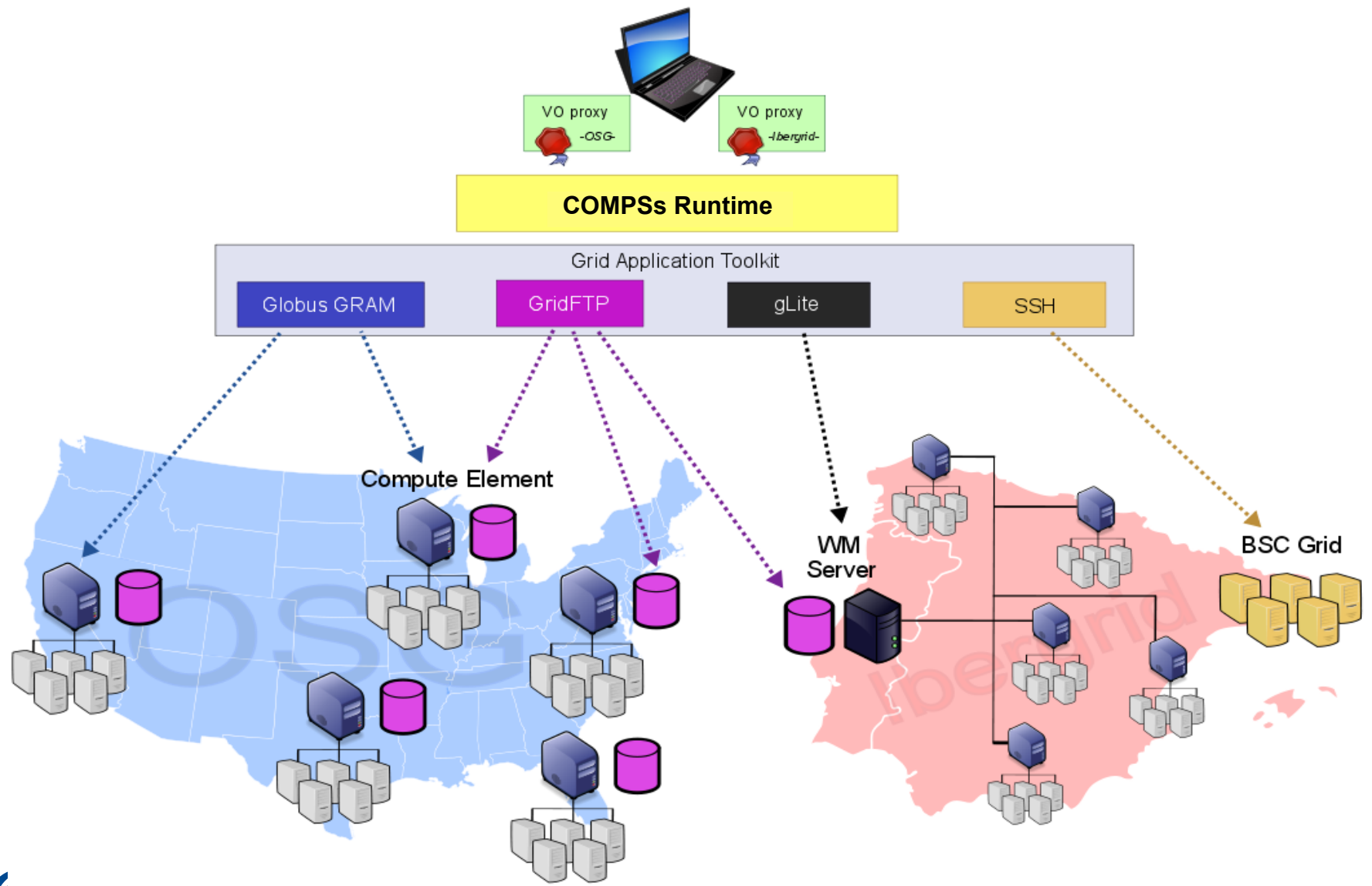
# COMPSs in a Cluster (queue system)

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



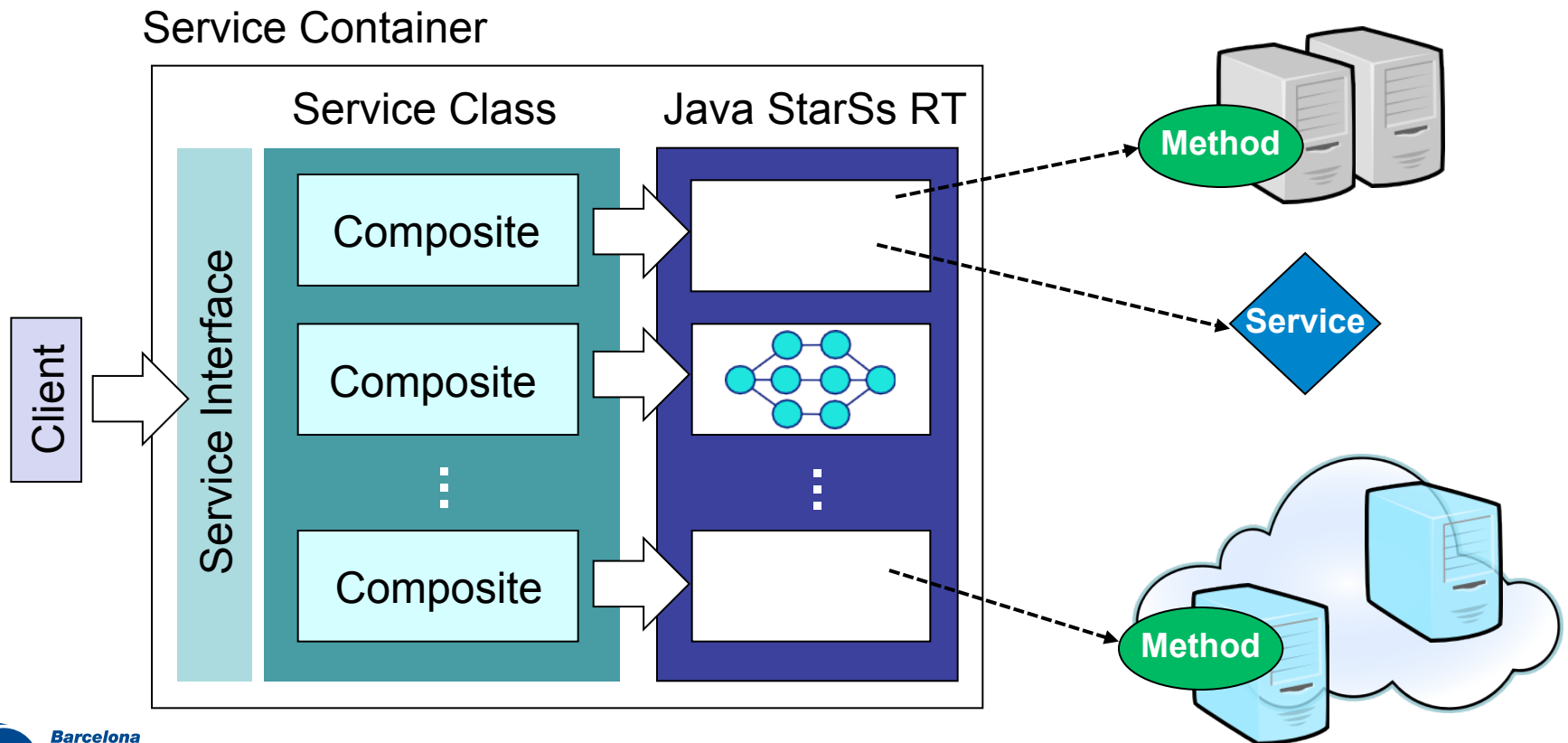


# COMPSs in a Grid



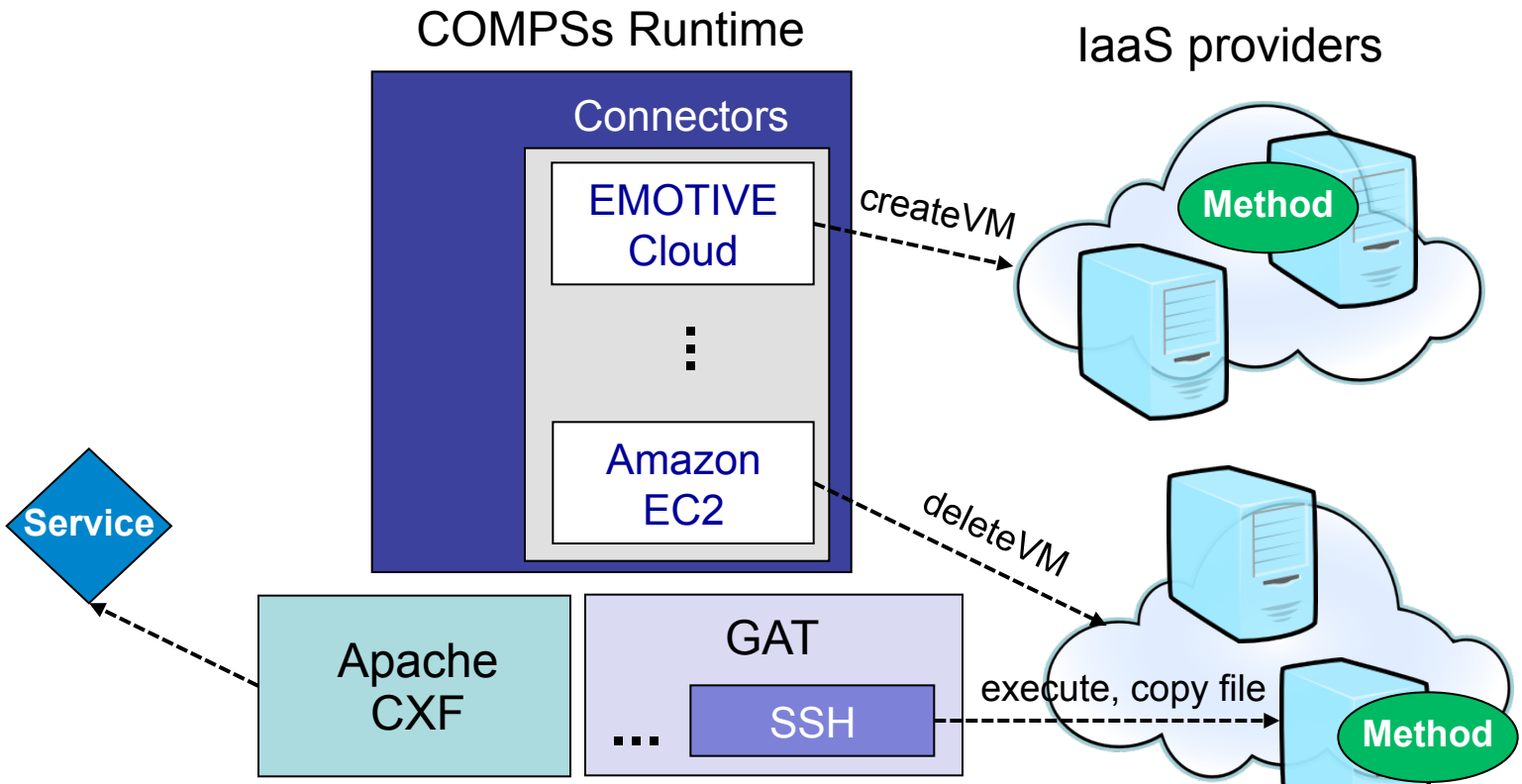
# COMPSs in the Cloud

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



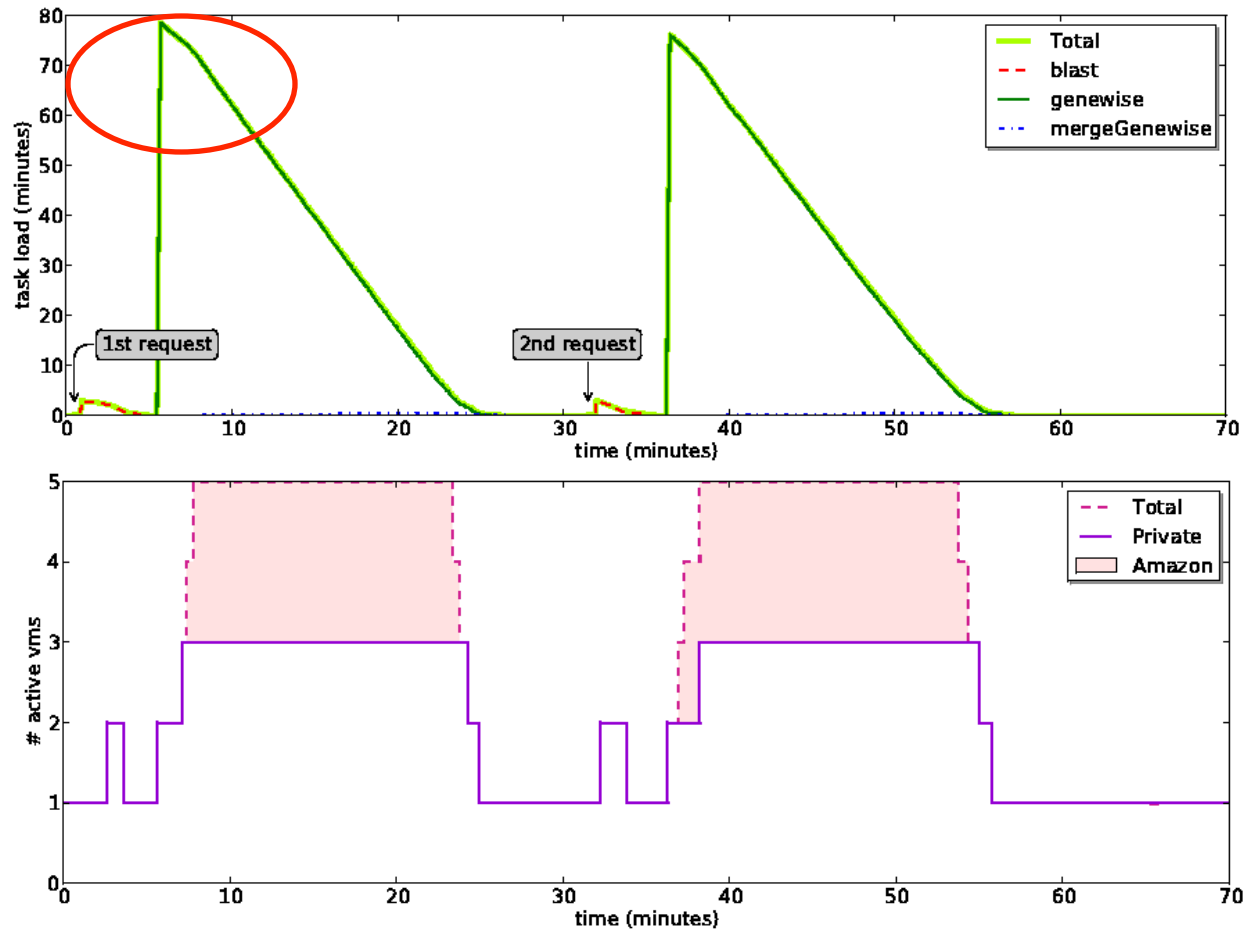
# Cloud: Connector design

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

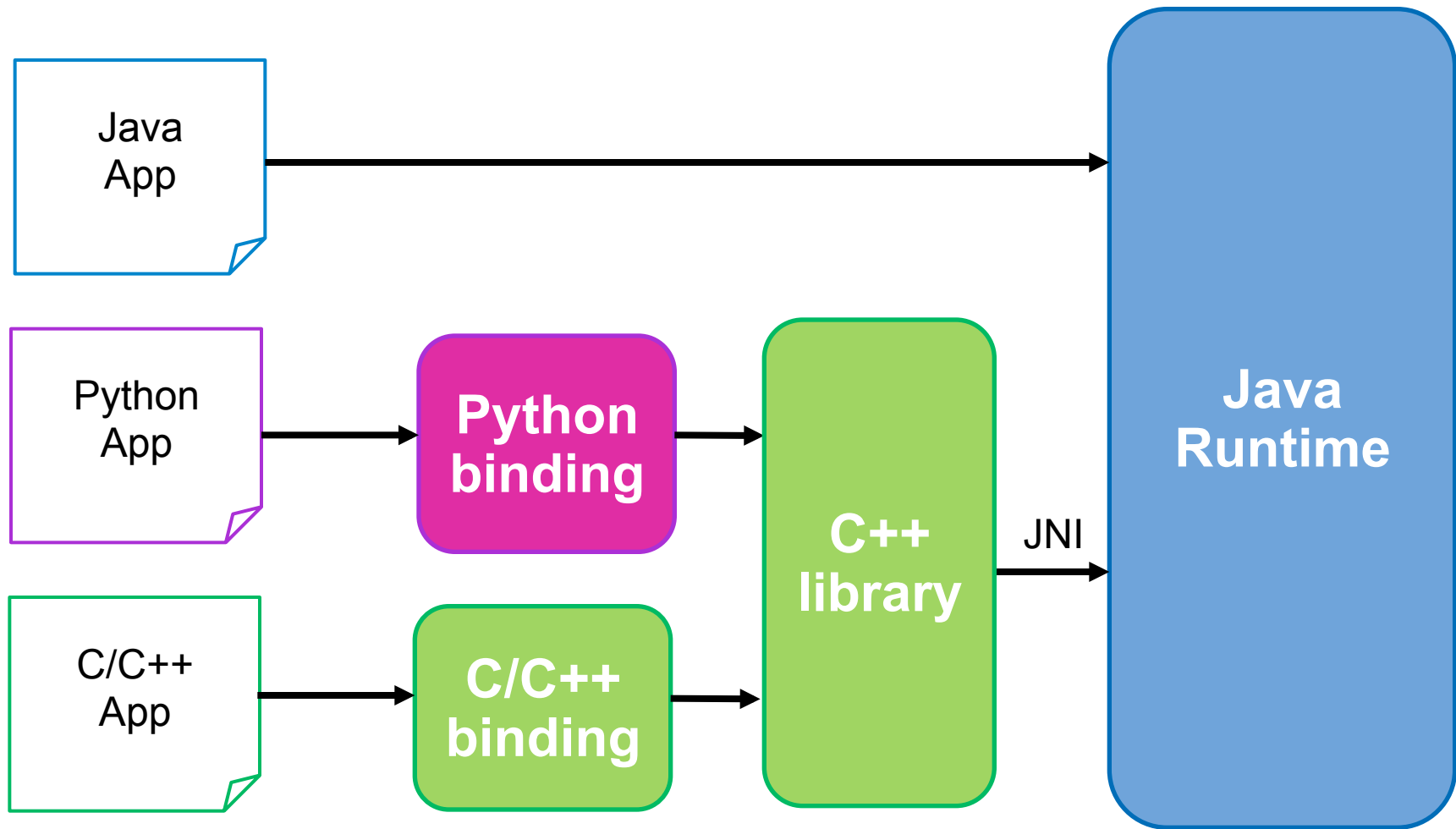


# Cloud: Elasticity

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



# COMPSs Bindings





**Barcelona  
Supercomputing  
Center**  
*Centro Nacional de Supercomputación*

**DEMOS**

# Matmul example

```
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 Matmultf {  
    @Constraints(processorCPUCount = 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
- javac matmul/src/matmul/\*.java
- cd matmul/src/
- 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
- cd
- cp ./matmul/src/matmul.jar .

## « In remote machines

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

# Matmul: Executing

## « Set CLASSPATH

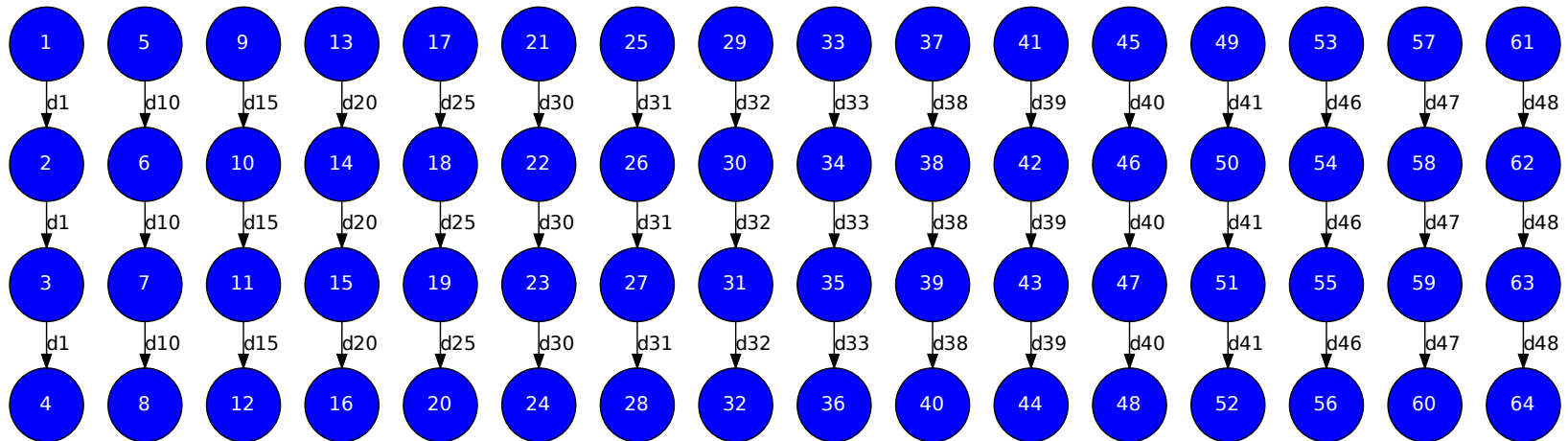
- export CLASSPATH=\$CLASSPATH:/home/user/matmul.jar
- runcompss matmul.Matmul 4

# Matmul: Monitoring execution

## « Browse

- <http://localhost:8080/compss-monitor>

- Blocks matrixes multiplication

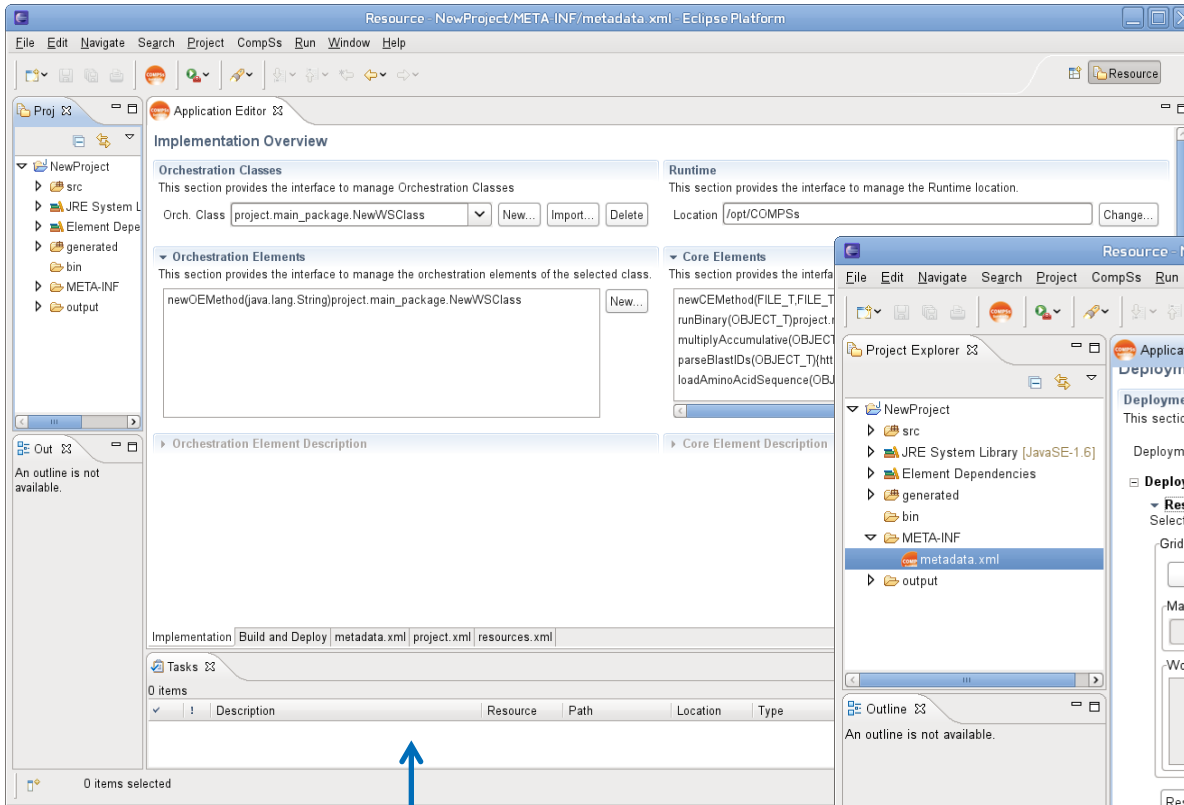


# IDE COMPSs applications as a Service

## IDE for implementing and deploying applications

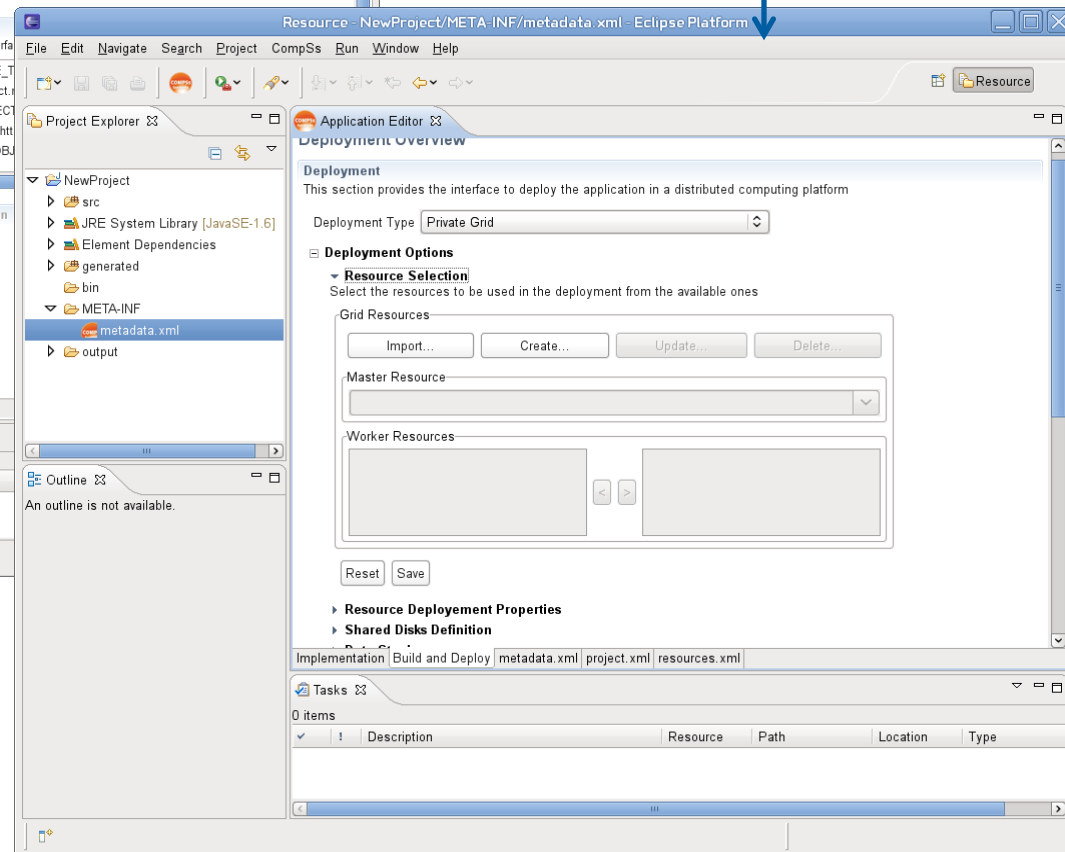
### Building & Deployment:

- Generate Packages
- Define hosts & Deploy



### Tasks Definition:

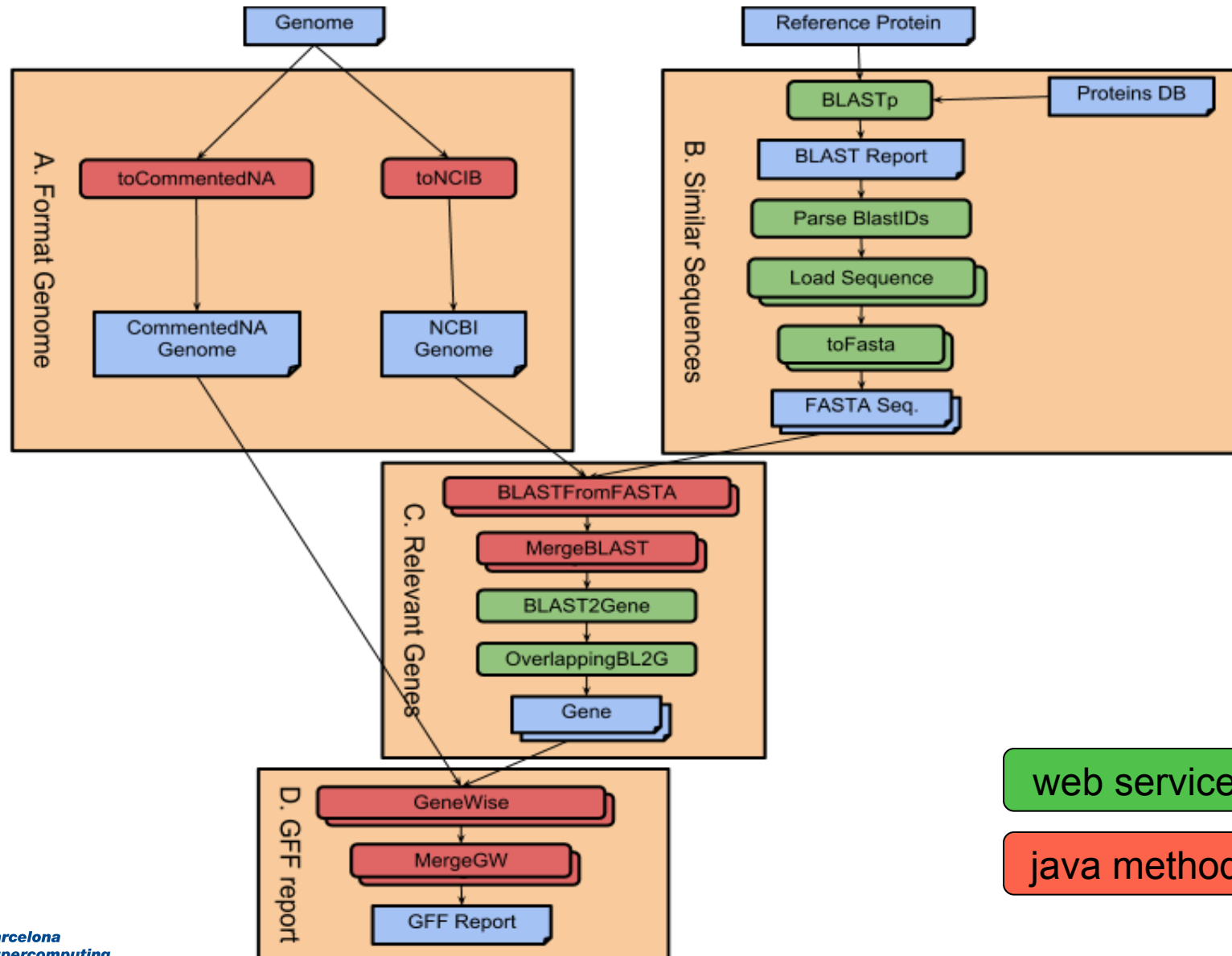
- Service Operations (Orchestration)
- Tasks (Core Element)



# 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







**Barcelona  
Supercomputing  
Center**

*Centro Nacional de Supercomputación*

**HANDS-ON**

# Hands-On: Overview

- COMPSs Virtual Machine setup
- Applications Overview (BLAST, HMMER, ...)
  - Code modification
  - Configuration, compilation & execution
  - Monitoring, debugging
  - Overview of tracing and trace performance analysis
  - IDE

# COMPSs development VM Installation

- **COMPSs Development & Test VM (64-bit) OVA**
  - **Available from USB**
- **Import the virtual appliance in VirtualBox**



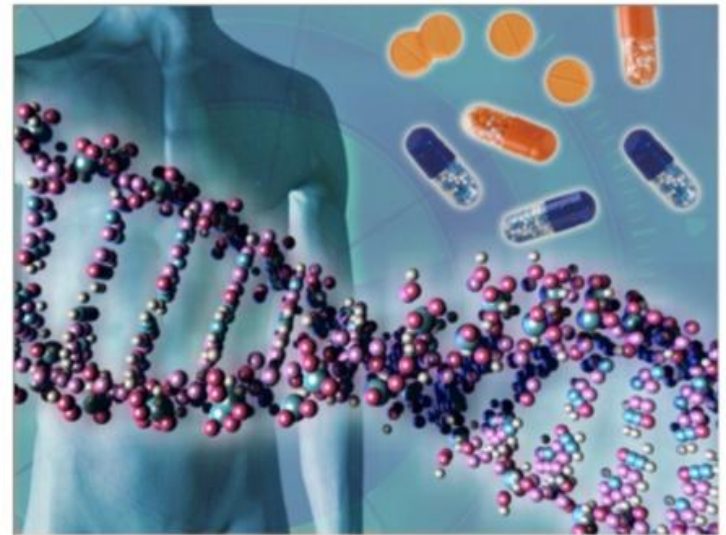
# BLAST: Hands-on



## Bioinformatics Scenario

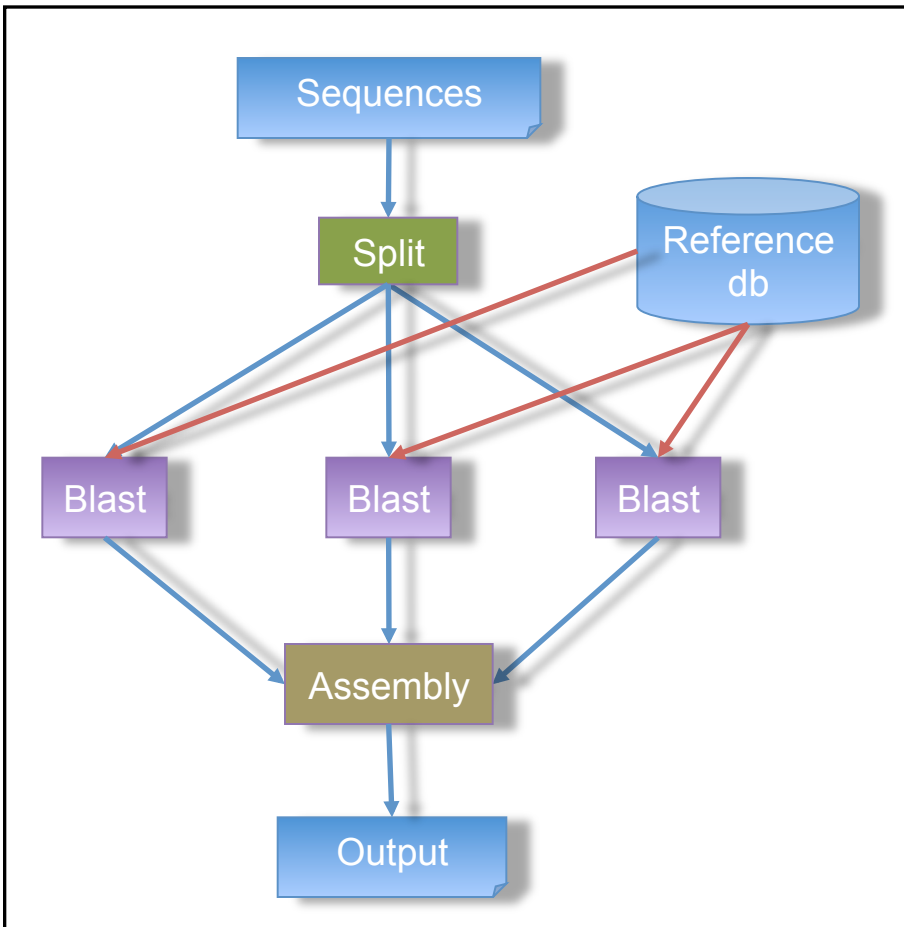
- « BLAST (Basic Local Alignment Search Tool) Suite:
  - BLAST: An algorithm for comparing primary biological sequence information, such as the amino-acid sequences of different proteins or nucleotides of DNA sequences.

*BLAST enables a researcher to compare a query sequence with a library or database of sequences, and identify sequences that resemble the query sequence above a certain threshold.*

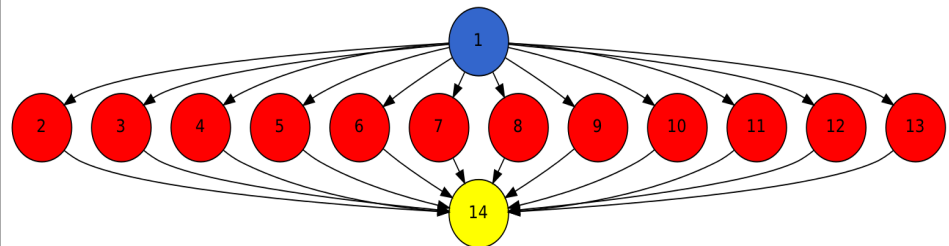


# BLAST: Hands-on

- **BLAST**

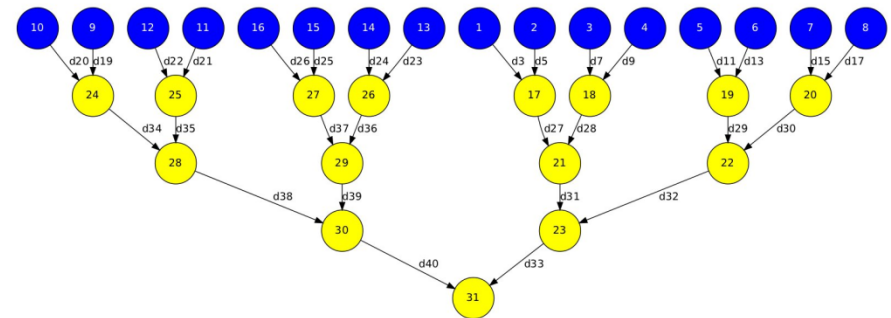


## All-to-One Reduction:



OR

## Tree-based Reduction:



# BLAST: All-to-One reduction

- Main Application (All-to-One):

```
public static void main(String args[]) throws Exception {  
  
    sequences[] = splitSequences(inputFile, nFrag);  
  
    for (partition: sequences)  
    {  
        BlastImpl.align(database, partition, partitionOutput, blastBinary, commandArgs);  
        partitionOutputs.add(partitionOutput);  
    }  
  
    assemblyPartitions(partialOutputs, outputFileName, tempDir, nFrag);  
}
```

# BLAST: All-to-One reduction

- Remote task implementation:

```
public class BlastImpl{  
  
    public void align(String databasePath, String partitionFile,  
                    String partitionOutput, String blastBinary, String commandArgs)  
    {  
        String cmd = blastBinary+ " " +"-p blastx -d " + databasePath + " -i " +partitionFile+ " -o "+  
                    partitionOutput+ " " +commandArgs;  
  
        Process simProc = Runtime.getRuntime().exec(cmd);  
        .....  
    }  
}
```



# BLAST: All-to-One reduction

Creation of the annotated interface for the selection of remote tasks

```
public interface BlastItf {  
  
    @Method(declaringClass = "blast.BlastImpl")  
    @Constraints(processorCPUCount = 4, memoryPhysicalSize = 4.0f)  
    void align(  
        @Parameter(type = Type.STRING, direction = Direction.IN)  
        String databasePath,  
  
        @Parameter(type = Type.FILE, direction = Direction.IN)  
        String partitionFile,  
  
        @Parameter(type = Type.FILE, direction = Direction.OUT)  
        String partitionOutput,  
  
        @Parameter(type = Type.STRING, direction = Direction.IN)  
        String blastBinary,  
  
        @Parameter(type = Type.STRING, direction = Direction.IN)  
        String commandArgs);  
  
}
```

# BLAST: Compilation and execution

- Compilation (Eclipse IDE)
  - *Package Explorer -> Project (blastallone) -> Export...*
- Usage
  - *runcompss blast.Blast <debug> <binary> <database> <sequences> <#fragments> <tmpdir> <output>*
- Execution
  - *cp ~/workspace/blastallone/jar/blast.jar ~*
  - *export CLASSPATH=\$CLASSPATH:/home/user/blast.jar*
  - *runcompss blast.Blast true /home/user/workspace/blast/binary/blastall /sharedDisk/Blast/databases/swissprot/swissprot /sharedDisk/Blast/sequences/sargasso\_test.fasta 4 /tmp/ /home/user/out.txt\_*

# BLAST: Compilation and execution

----- Executing blast.Blast in IT mode total-----

```
...
BLAST Sequence Alignment Tool:

Parameters:
- Debug Enabled
- Blast binary: /home/user/workspace/blastAllOne/binary/blastall
- Number of expected fragments: 8
- Database Name with Path: /sharedDisk/Blast/databases/swissprot/swissprot
- Database Name: swissprot
- Input Sequences File: /sharedDisk/Blast/sequences/sargasso_test.fasta
- Temporary Directory: /tmp/
- Output File: /home/user/IT/blast.Blast/out.txt
- Command Line Arguments:

- The total number of sequences is: 20
- The total number of sequences of a fragment is: 3

- Splitting sequences among fragment files...
[ API] - Opening file /tmp/seqFile1b495168-e913-430a-a347-9894015911e1.sqf in mode APPEND
...

Aligning Sequences:

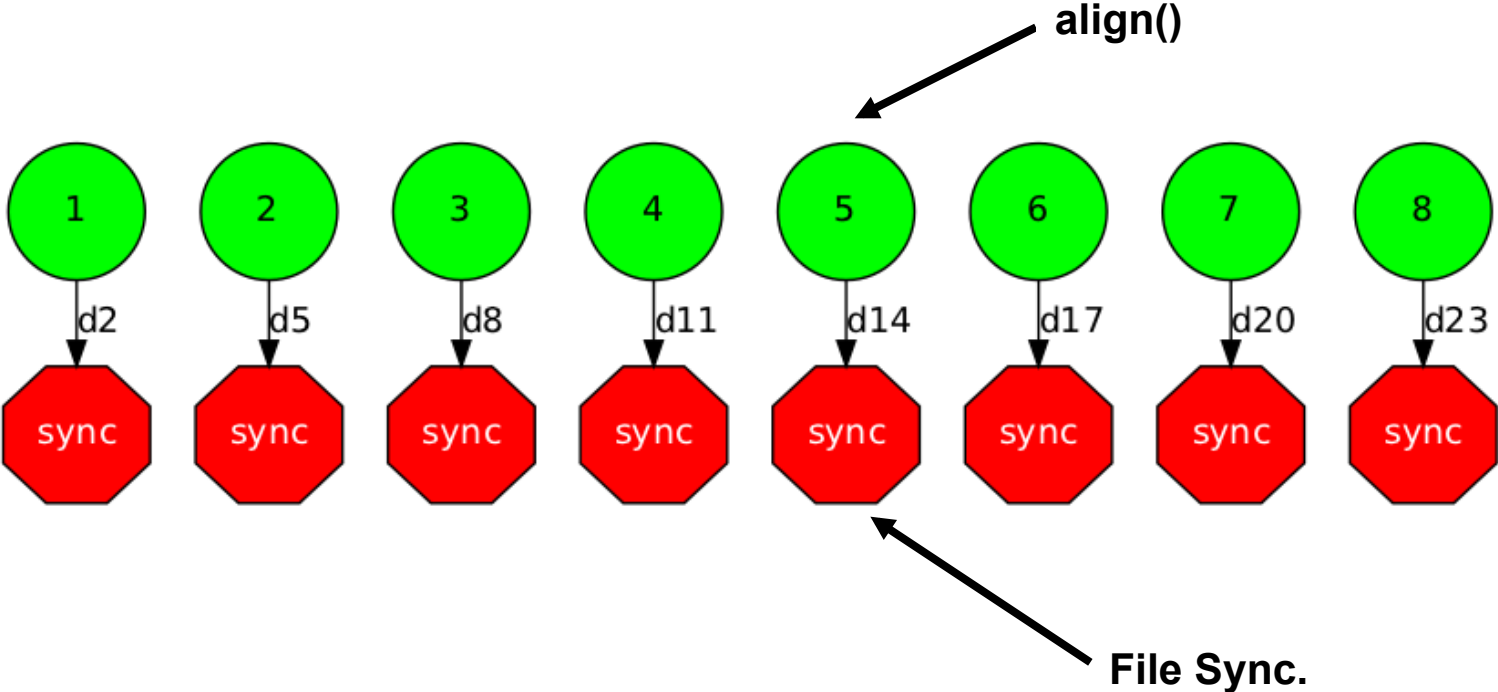
- Number of fragments to assemble -> 8

[ API] - Opening file /home/user/IT/blast.Blast/out.txt in mode WRITE
- Assembling partial output -> /tmp/resFile1b495168-e913-430a-a347-9894015911e1.result.txt to final output file -> /home/user/IT/blast.Blast/out.txt
...
- Assembling partial output -> /tmp/resFile270855af-307b-4a1e-bc42-0e0cf22256ae.result.txt to final output file -> /home/user/IT/blast.Blast/out.txt
-Sequences assembled in 184 seconds
...
-----
```

# BLAST: All-to-One (work)

- Generate the final graph
- Launch BLAST (All-to-One)

# BLAST: All-to-One (Graph)



# BLAST: Tree-based reduction (work)

- Code the final reduction and its interface.

# BLAST: Tree-based reduction

- Main Application (Tree-based):

```
public static void main(String args[]) throws Exception {  
  
    sequences[] = splitSequences(inputFile, nFragments);  
  
    for (partition: sequences)  
    {  
        BlastImpl.align(database, partition, partitionOutput, blastBinary, commandArgs);  
        partitionOutputs.add(partitionOutput);  
    }  
  
    //Final Assembly process -> Merge 2 by 2  
    int neighbour=1;  
    while (neighbour<partialOutputs.size()){  
        for (int result=0; result<partialOutputs.size(); result+=2*neighbour){  
            if (result+neighbour < partialOutputs.size()){  
                BlastImpl.assemblyPartitions(partialOutputs.get(result),partialOutputs.get(result+neighbour));  
                lastMerge = partialOutputs.get(result);  
            }  
        }  
        neighbor*=2;  
    }  
}
```

# BLAST: Tree-based reduction

Creation the annotated interface for the selection of the remote tasks

```
public interface BlastItf {  
  
    @Method(declaringClass = "blast.BlastImpl")  
    @Constraints(processorCPUCount = 4, memoryPhysicalSize = 4.0f)  
    void align(  
        @Parameter(type = Type.STRING, direction = Direction.IN)  
        String databasePath,  
  
        @Parameter(type = Type.FILE, direction = Direction.IN)  
        String partitionFile,  
  
        ....  
  
        @Parameter(type = Type.STRING, direction = Direction.IN)  
        String commandArgs);  
  
    @Method(declaringClass = "blast.BlastImpl")  
    @Constraints(processorCPUCount = 2, memoryPhysicalSize = 2.0f)  
    void assemblyPartitions(  
        @Parameter(type = Type.FILE, direction = Direction.INOUT)  
        String partialFileA,  
  
        @Parameter(type = Type.FILE, direction = Direction.IN)  
        String partialFileB);  
}
```



# BLAST: Tree-based execution

----- Executing blast.Blast in IT mode total-----

...

BLAST Sequence Alignment Tool:

Parameters:

- Debug Enabled
- Blast binary: /home/user/workspace/blastAllOne/binary/blastall
- Number of expected fragments: 8
- Database Name with Path: /sharedDisk/Blast/databases/swissprot/swissprot
- Database Name: swissprot
- Input Sequences File: /sharedDisk/Blast/sequences/sargasso\_test.fasta
- Temporary Directory: /tmp/
- Output File: /home/user/IT/blast.Blast/out.txt
- Command Line Arguments:

- The total number of sequences is: 20
- The total number of sequences of a fragment is: 3

- Splitting sequences among fragment files...

[ API] - Opening file /tmp/seqFileb0fa2b12-d0f6-42c1-b499-1e207e30ad84.sqf in mode APPEND

...

Aligning Sequences:

- **Number of fragments to assemble -> 8**

- **Merging files -> /tmp/resFileb0fa2b12-d0f6-42c1-b499-1e207e30ad84.result.txt and /tmp/resFile815b4ff6-a077-422b-bc9b-9c6e10d8a417.result.txt**

...

- **Merging files -> /tmp/resFileb0fa2b12-d0f6-42c1-b499-1e207e30ad84.result.txt and /tmp/resFile81605bf8-b0f4-46bc-a521-9f289d219ef3.result.txt**

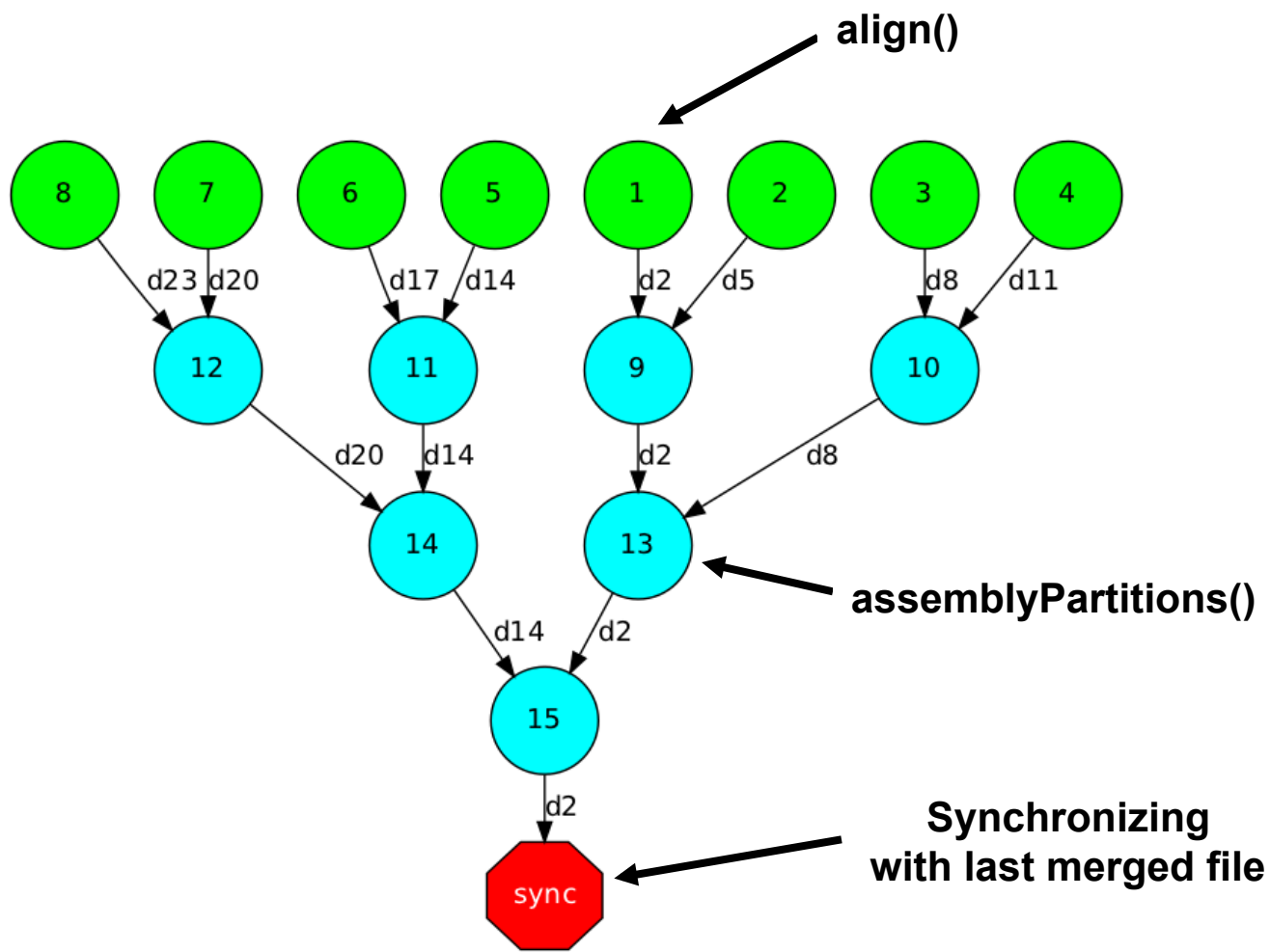
**Moving last merged file: /tmp/resFileb0fa2b12-d0f6-42c1-b499-1e207e30ad84.result.txt to /home/user/IT/blast.Blast/out.txt**

[ API] - Opening file /home/user/IT/blast.Blast/out.txt in mode WRITE

- /sharedDisk/Blast/sequences/sargasso\_test.fasta sequences aligned successfully in 193 seconds

...

# BLAST: Tree-based (Graph)



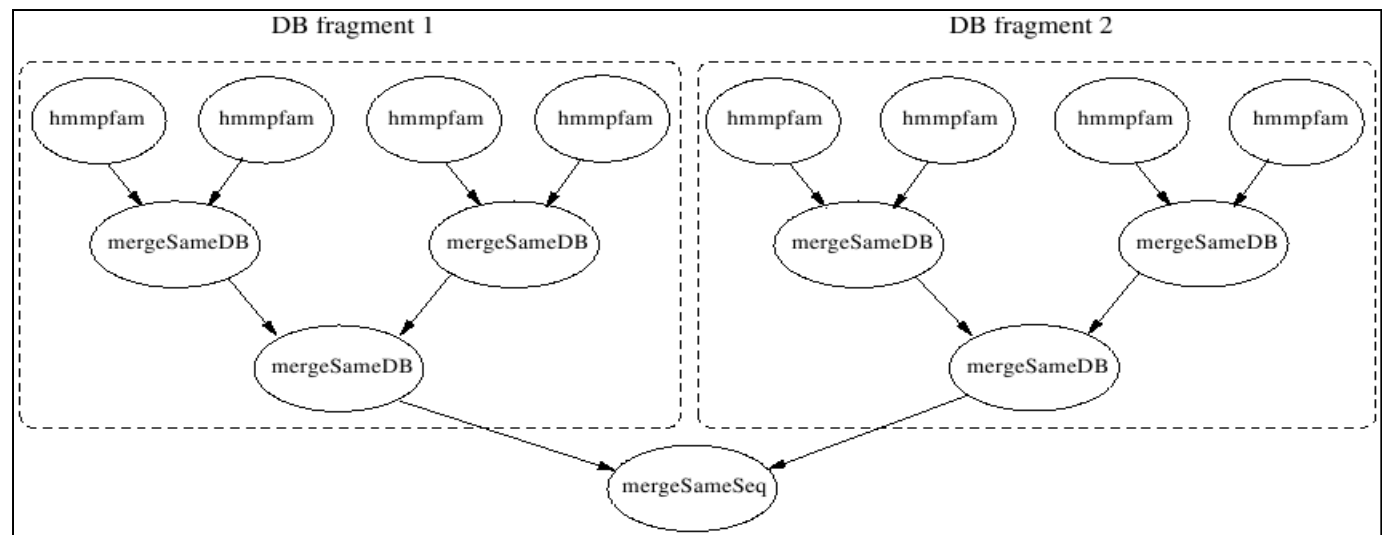
# HMMER: Hands-on



# HMMER 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 service
  - Starting point: sequential version of the hmmpfam tool
- 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

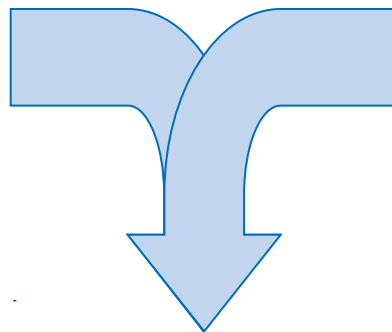


# HMMER example

## HMMER

Protein Database

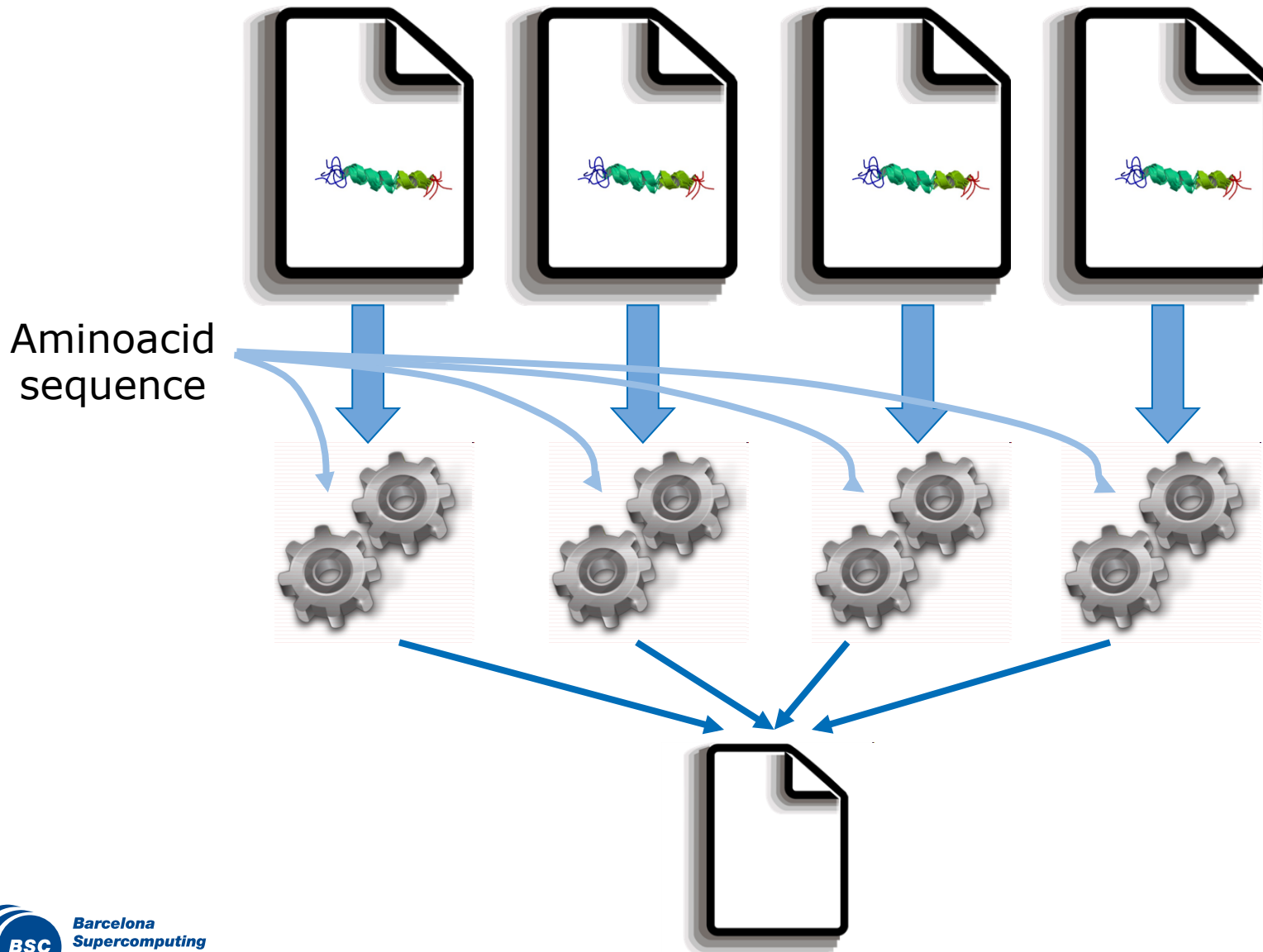
Aminoacid Sequence



IQKKSGKWHTLTDLRA  
VNAVIQPMGPLQPGLP  
SPAMIPKDWPLIIDLK  
DCFFTIPLAEQDCEKFA  
FTIPAINNKEPATRF

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 example



# HMMER example (code)

```
String[] outputs = new String[numDBFragments];

//Process
for (String dbFrag : dbFragments) {
    outputs[dbNum]= HMMPFamImpl.hmmpfam(sequence, dbFrag);
}

//Merge all DB outputs of the same DB fragment
int neighbour = 1;
while (neighbour < numDBFragments) {
    for (int db = 0; db < numDBFragments; db += 2 * neighbour) {
        if (db + neighbour < numDBFragments) {
            HMMPFamImpl.mergeSameDB(outputs[db], outputs[db + neighbour]);
        }
    }
    neighbour *= 2;
}
```

# HMMER example (code)

```
public interface HMMPfamItf {
```

```
    @Method(declaringClass = "worker.hmmerobj.HMMPfamImpl")
```

```
    @Constraints(storageElemSize = 1.5f)
```

```
    String hmmpfam(
```

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

```
        String seqFile,
```

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

```
        String dbFile,
```

```
        ...
```

```
    );
```

```
    @Method(declaringClass = "worker.hmmerobj.HMMPfamImpl")
```

```
    void mergeSameDB(
```

```
        @Parameter(type = Type.OBJECT, direction = Direction.IN)
```

```
        String resultFile1,
```

```
        @Parameter(type = Type.OBJECT, direction = Direction.IN)
```

```
        String resultFile2
```

```
    );
```

```
    ...
```

```
}
```

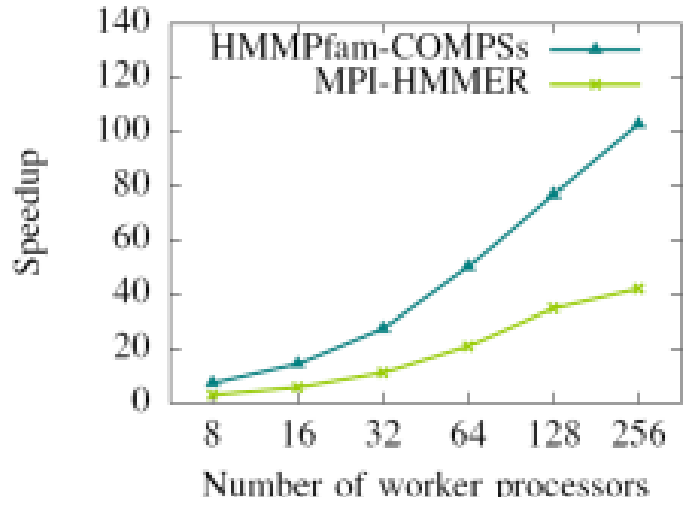
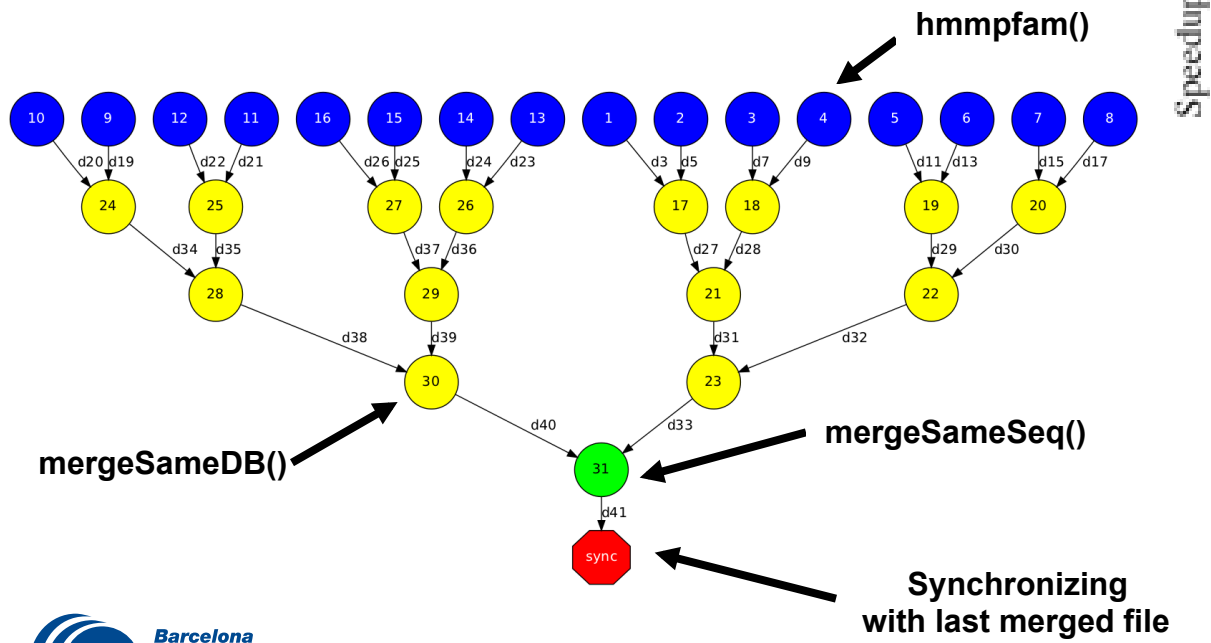
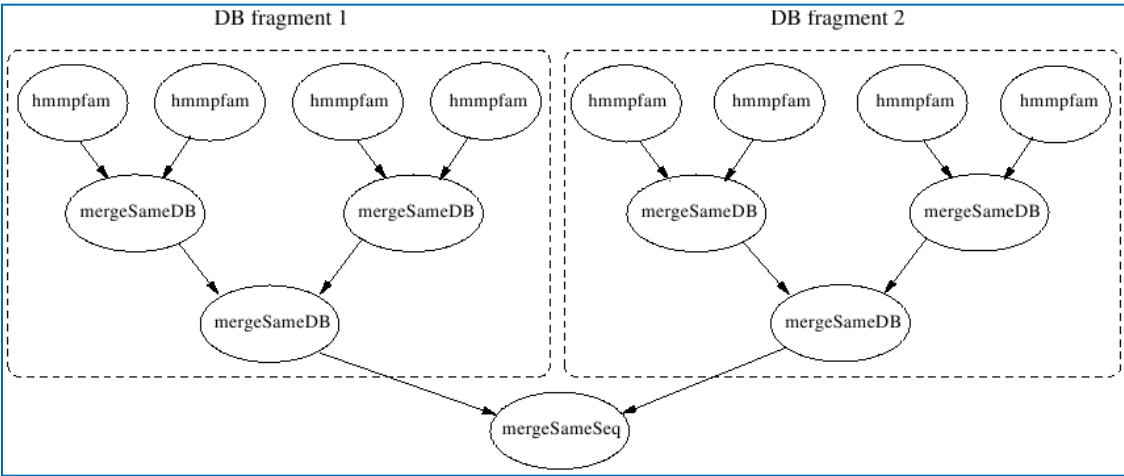
Implementation

Task constraints

Parameter metadata



# HMMER example (workflow)



# HMMER: Task Selection (work)

- Complete the hmmpfam & mergeSameSeq method interfaces.

# 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/</InstallDir>
    <WorkingDir>/tmp/</WorkingDir>
    <User>user</User>
    <LimitOfTasks>2</LimitOfTasks>
  </Worker>
</Project>
```

# HMMER: Configuration, compilation and execution

- Configuration: /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>
        <CPUCount>2</CPUCount>
      </Processor>
      <OS>
        <OSType>Linux</OSType>
        <MaxProcessesPerUser>32</MaxProcess
      </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
```

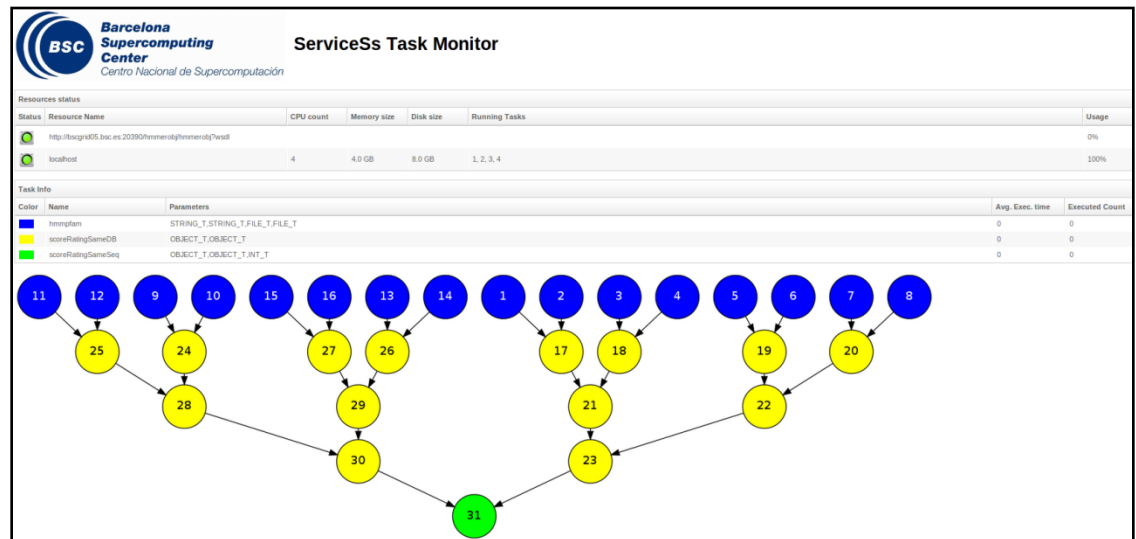
```
----- Executing hmmerobj.HMMPfam in IT mode total-----
```

```
[ 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: Monitoring

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



# HMMER: 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
    - `~/IT/[APP_NAME]/jobs/jobN.[out|err]`
  - Messages from the runtime COMPSs
    - `~/it.log`
  - Task to resources allocation:
    - `~/resources.log`
- The user can verify the correct structure of the parallel application generating a complete post-mortem application graph
  - `gengraph $HOME/APP_NAME.dot`



# 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**
  - **Instrumentation and Visualization tools from BSC are needed.**

# 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>

# Tracing: Instrumentation

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

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

← COMPSs runtime starts

Welcome to **Extræ 2.4.3rc4 (revision 311 based on framework/trunk/files/extræ)**

← Extræ starts before the user application execution

- Extræ: Generating intermediate files for Paraver traces.**
- Extræ: Intermediate files will be stored in /home/user/IT/hmmerobj.HMMPfam**
- Extræ: Tracing buffer can hold 500000 events**
- Extræ: Tracing mode is set to: Detail.**
- Extræ: Successfully initiated with 1 tasks**

[ API] - Ready to process tasks

...  
...  
...

Extræ keeps tracing events in background



# Tracing: Instrumentation

[ API] - No more tasks for app 1  
[ API] - Stopping IT  
[ API] - Cleaning

**Extrae: Application has ended. Tracing has been terminated.**

...

**merger: Output trace format is: Paraver**

**merger: Extrae 2.4.3rc4 (revision 311 based on framework/trunk/files/extrae)**

...

[ API] - Integrated Toolkit stopped

...

**mpi2prv: Selected output trace format is Paraver**

**mpi2prv: Parsing intermediate files**

**mpi2prv: Generating tracefile (intermediate buffers of 1342156 events)**

**mpi2prv: Congratulations! hmmerobj.HMMPfam\_compss\_trace\_1392736225.prv has been generated.**

---

The application finishes and the tracing process ends

The merge process starts

COMPSs runtime ends

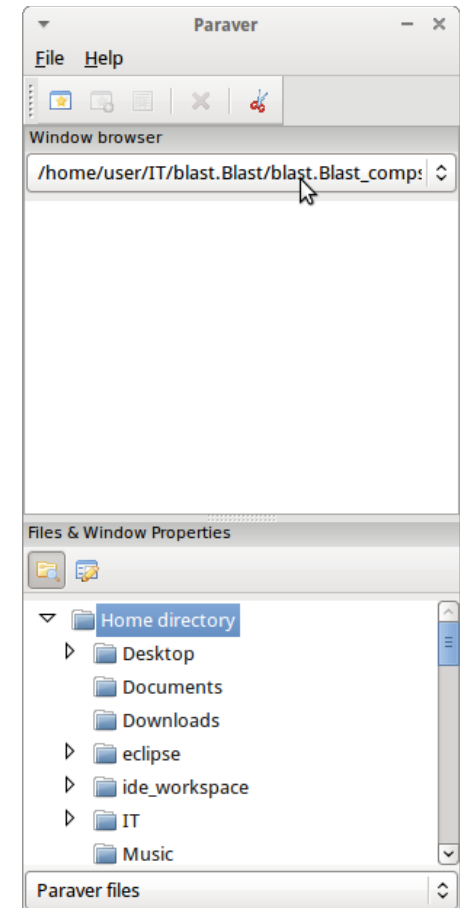
Intermediate trace files are processed

The final trace file is generated

# 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>

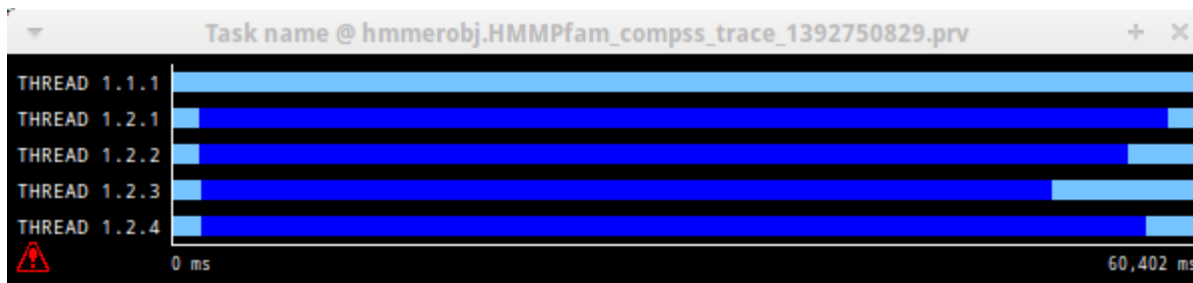


# Tracing: Hands-on

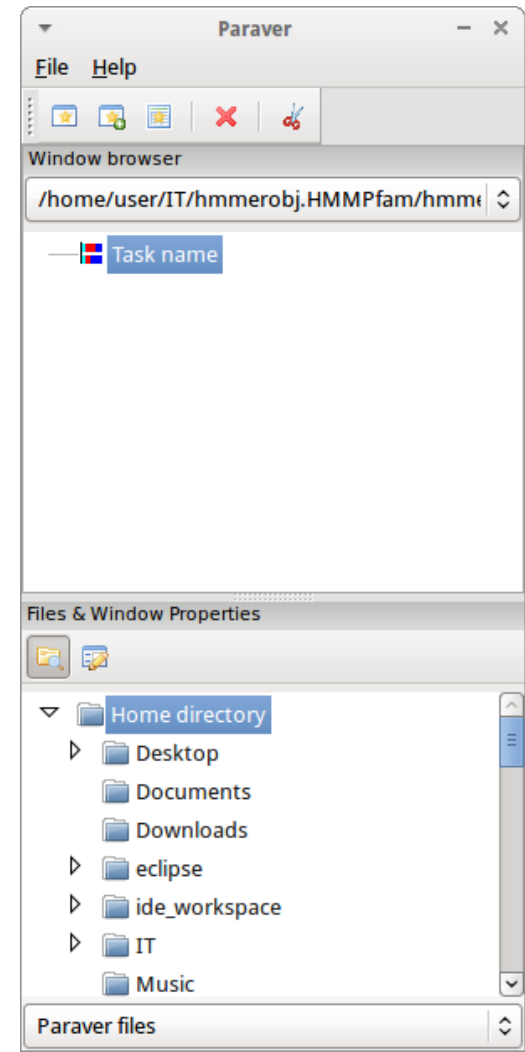
- Compilation (Eclipse IDE)
  - *Package Explorer -> Project (hmmerobj) -> Export...*
- Execution
  - *cp ~/workspace/hmmerobj/jar/hmmerobj.jar ~*
  - *export CLASSPATH=\$CLASSPATH:/home/user/hmmerobj.jar*
  - *runcompssext --app=hmmerobj.HMMPfam --tracing=true --cline\_args="/sharedDisk/Hmmer/smart.HMMs.bin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

# Tracing: Hands-on

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

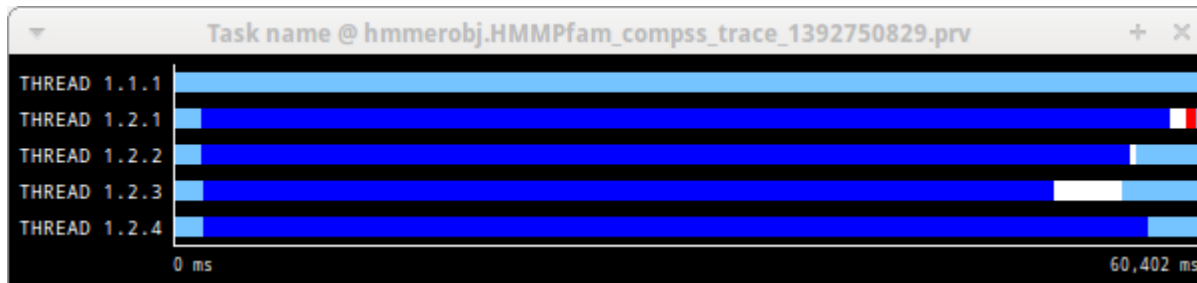


- Some small adjustments must be done in order to view the trace correctly

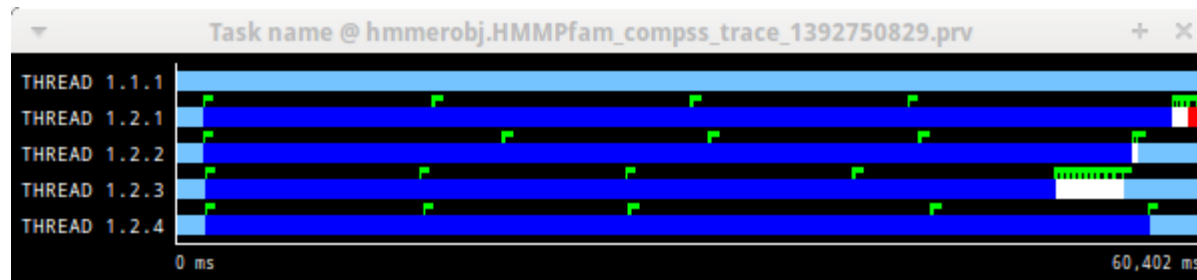


# Tracing: Hands-on

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



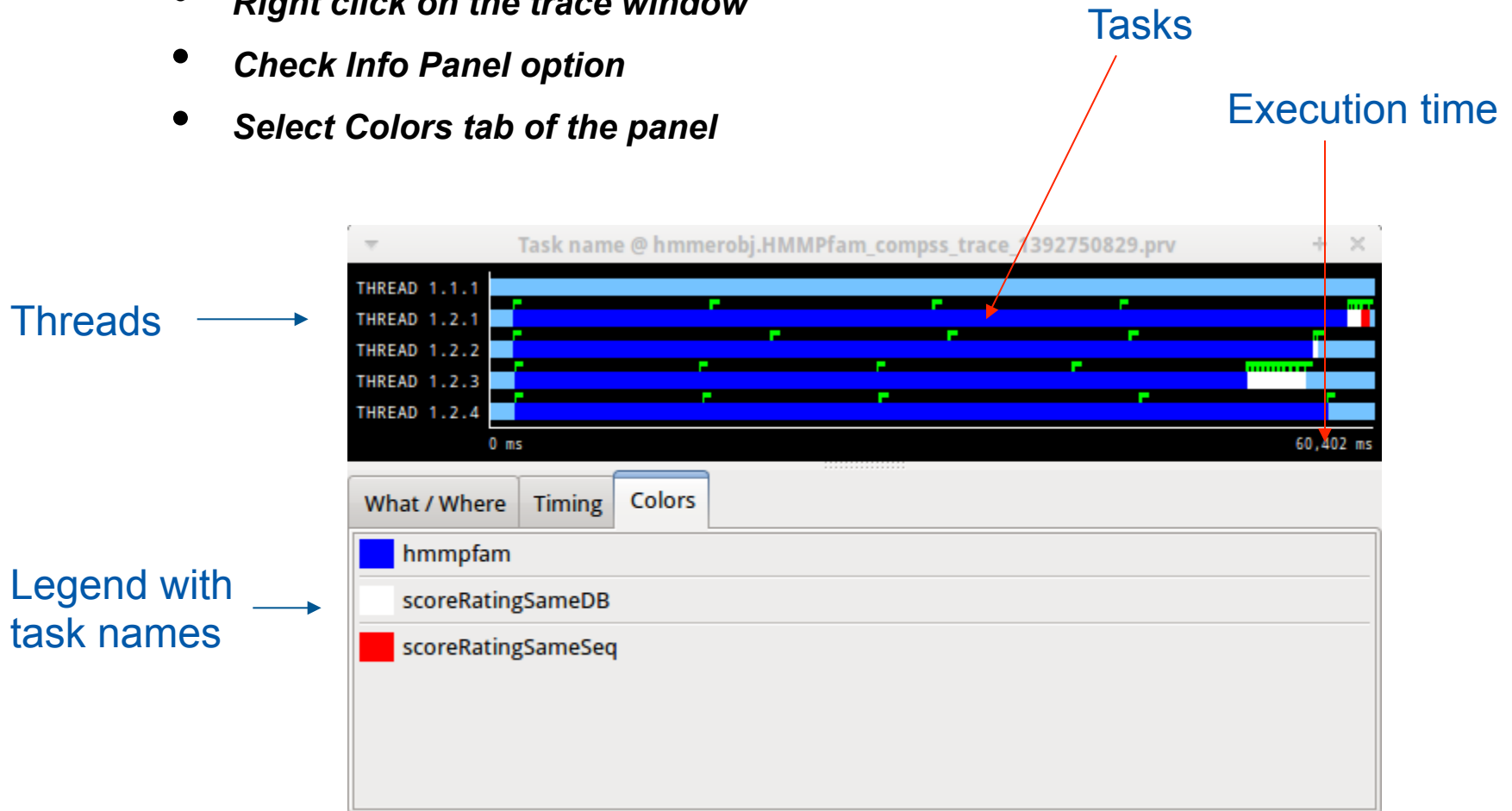
- View Event Flags
  - *Right click on the trace window*
  - *View / Event Flags*





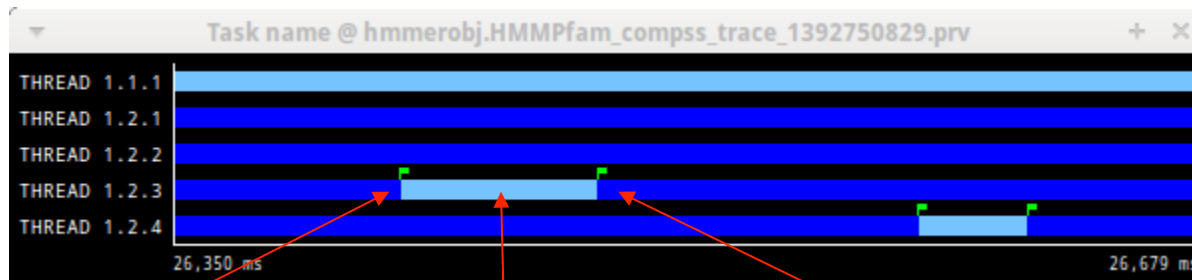
# Tracing: Hands-on

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



# Tracing: Hands-on

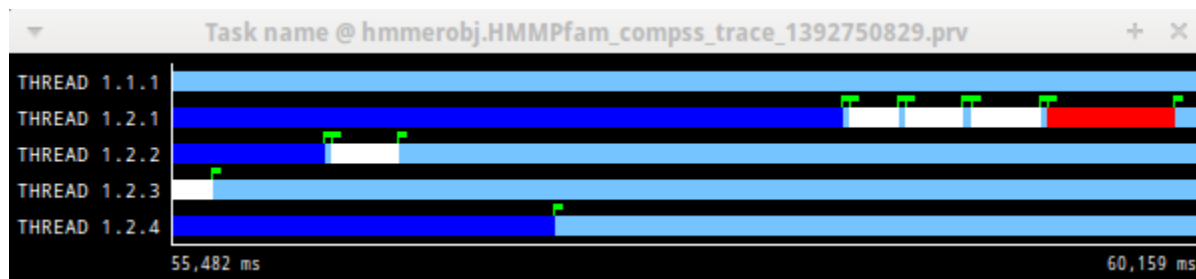
- Zoom to see details
  - *Select a region in the trace window to see in detail*
  - *And repeat the process til the needed zoom level*
  - *The undo zoom option is in the right click panel*



Previous task ends

Processor is idle

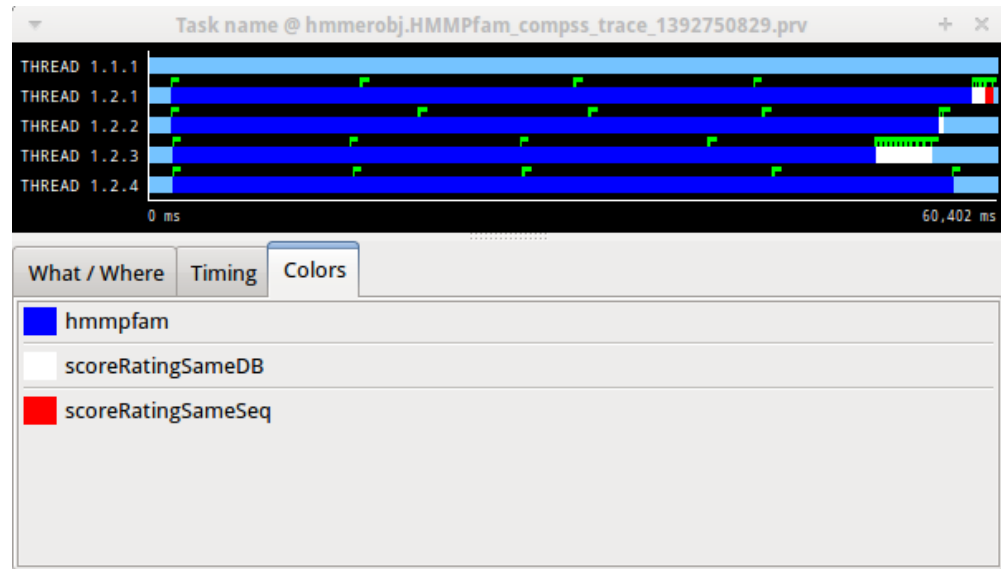
New task starts



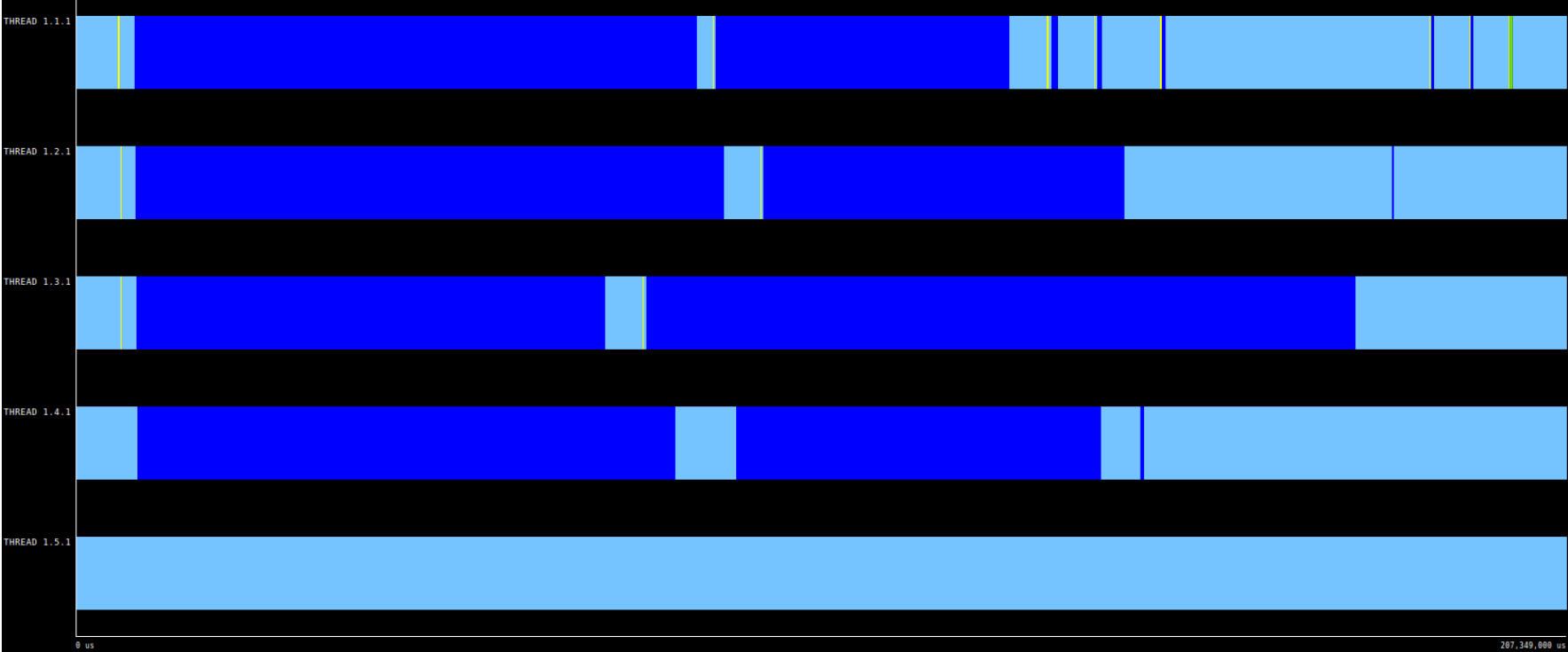
# Tracing: Hands-on

## 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



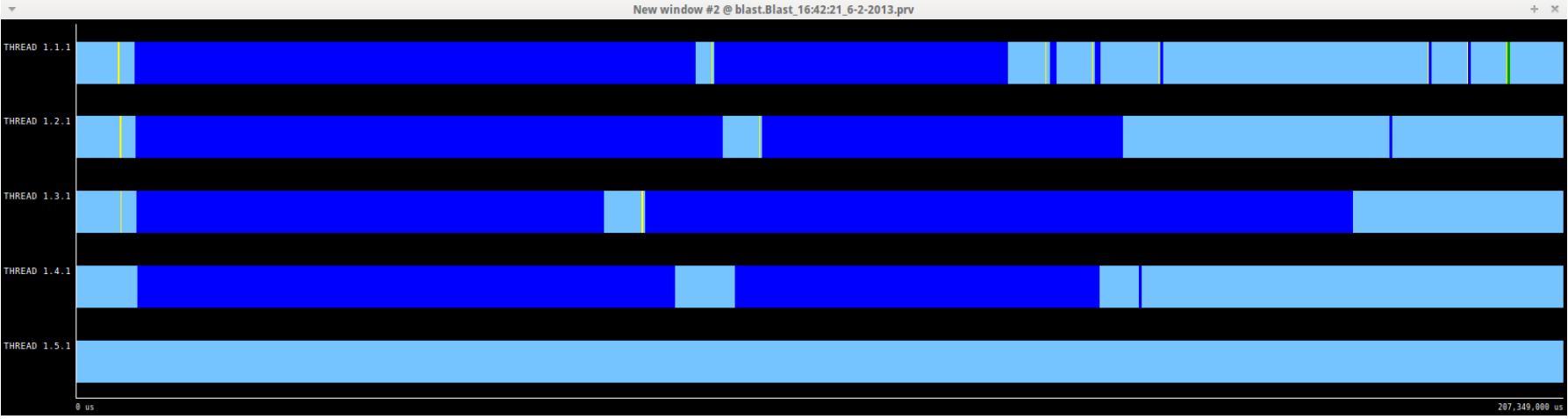
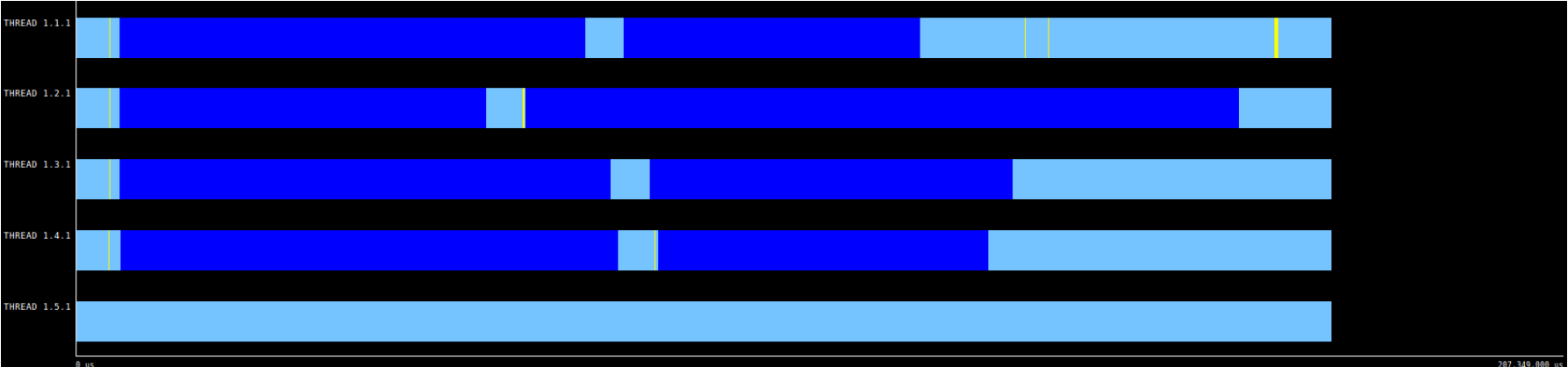
# Tracing: BLAST, Tree-based



What / Where Timing Colors

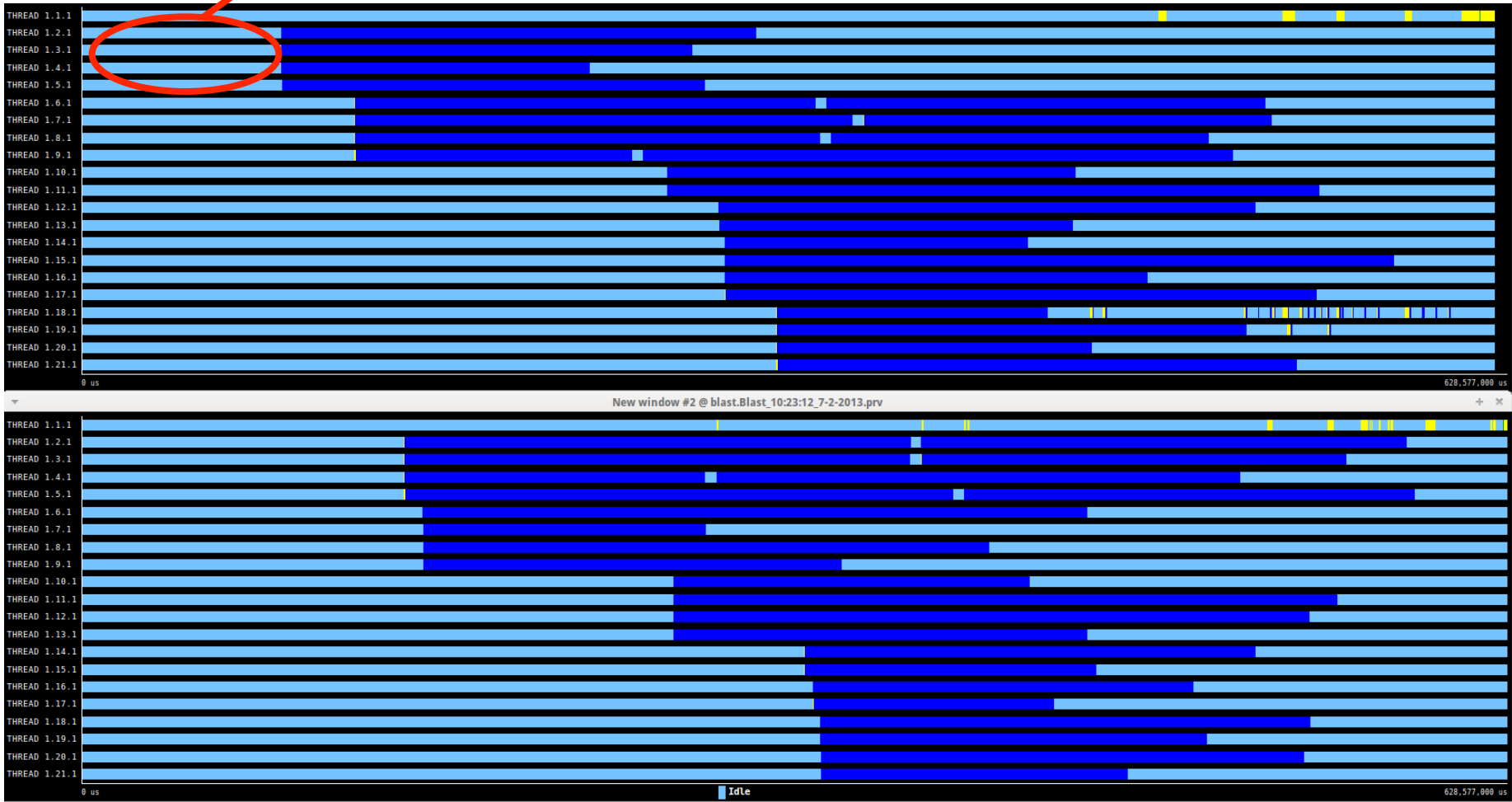
- Idle
- Transferring
- Running
- Waiting for transfer
- UR transferring

# Tracing: BLAST, All-to-One vs Tree-based (Local)

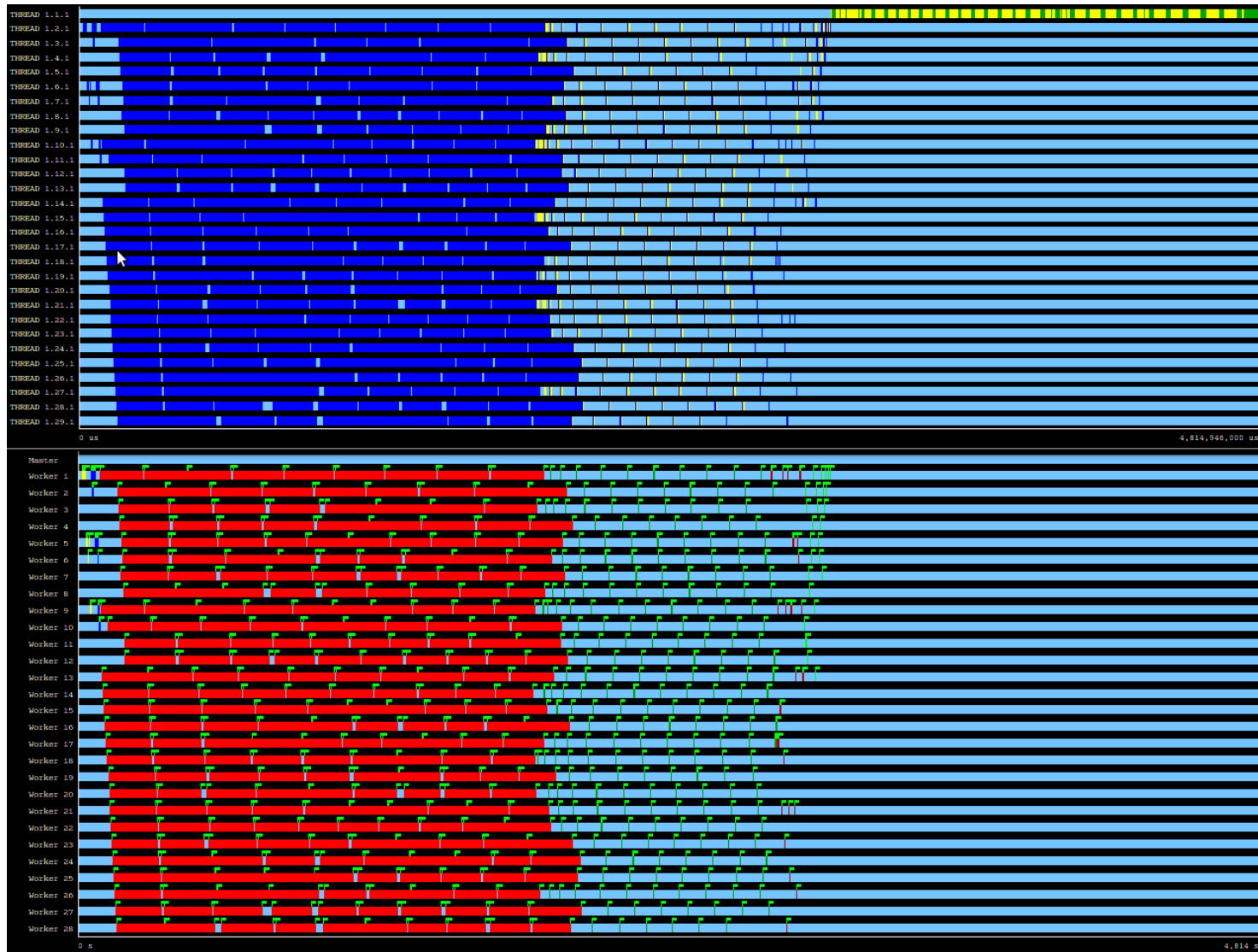


# Tracing: BLAST, All-to-One vs Tree-based (Cloud)

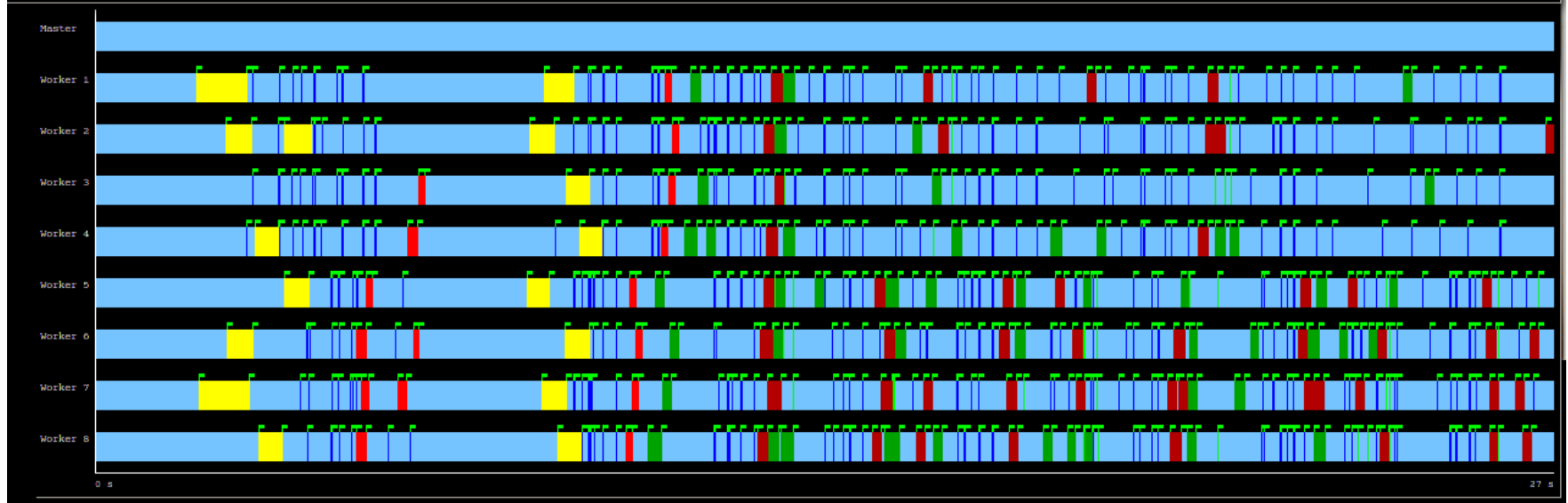
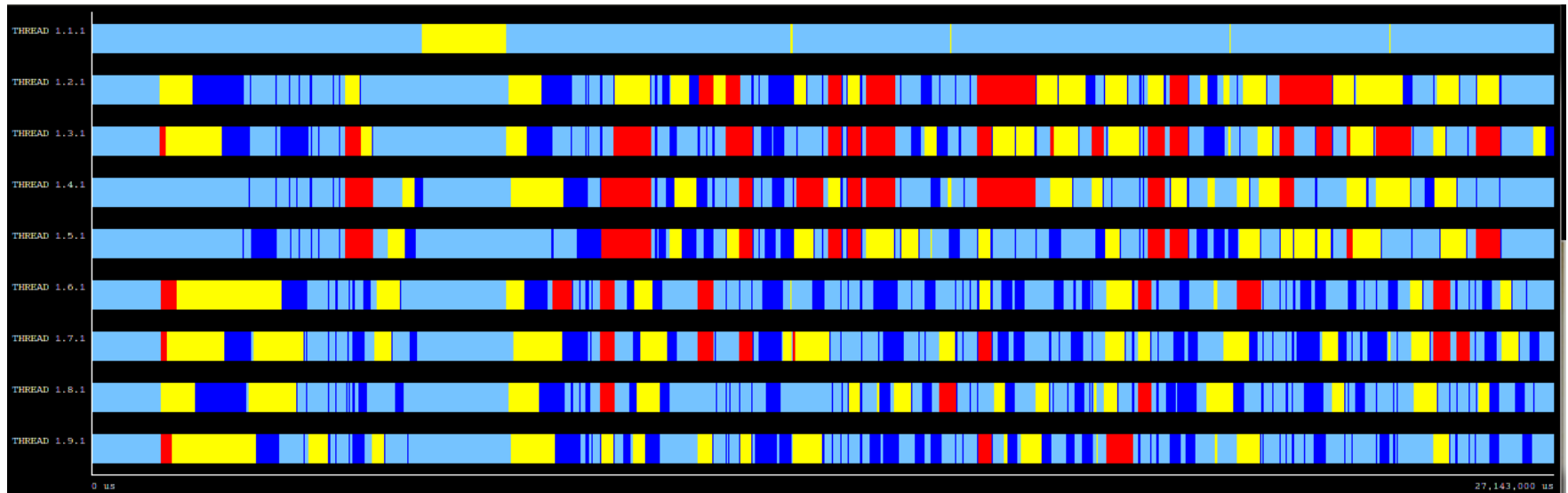
Virtual Machine Creation



# Tracing: Other examples



# Tracing: Other examples





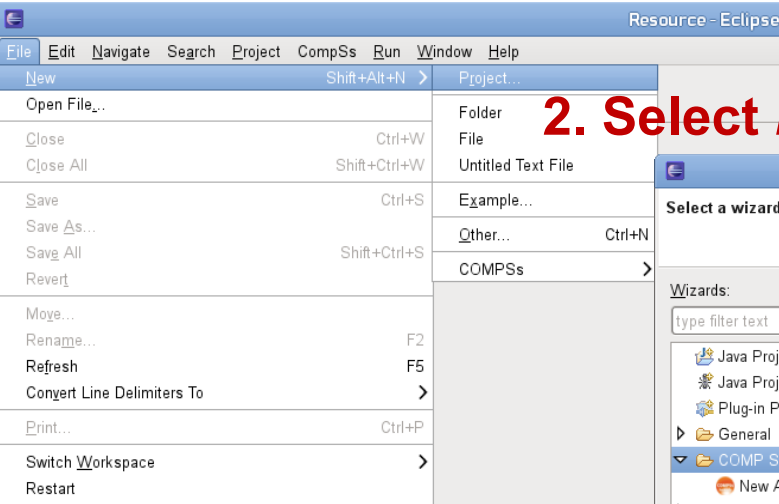


**Barcelona  
Supercomputing  
Center**  
*Centro Nacional de Supercomputación*

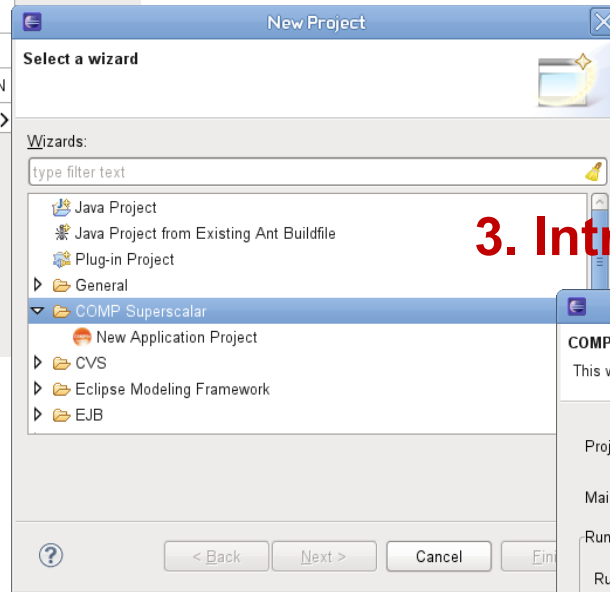
# Integrated Development Environment HANDS-ON

# IDE Hands On – Create a COMPSs Project

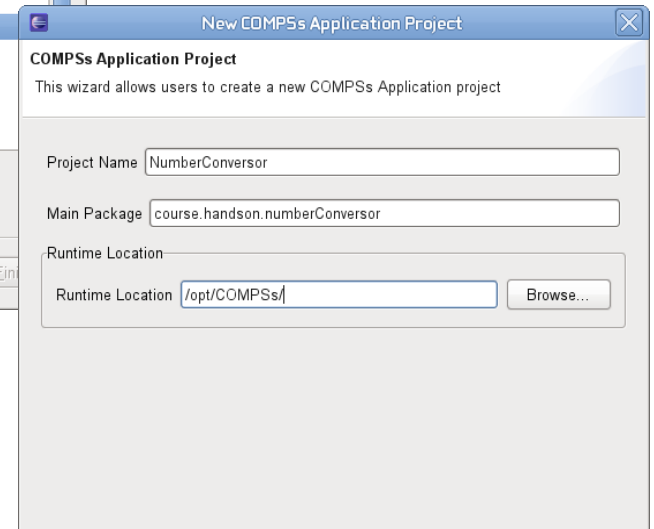
## 1. Menu *File->New -> Project...*



## 2. Select *New Application Project*



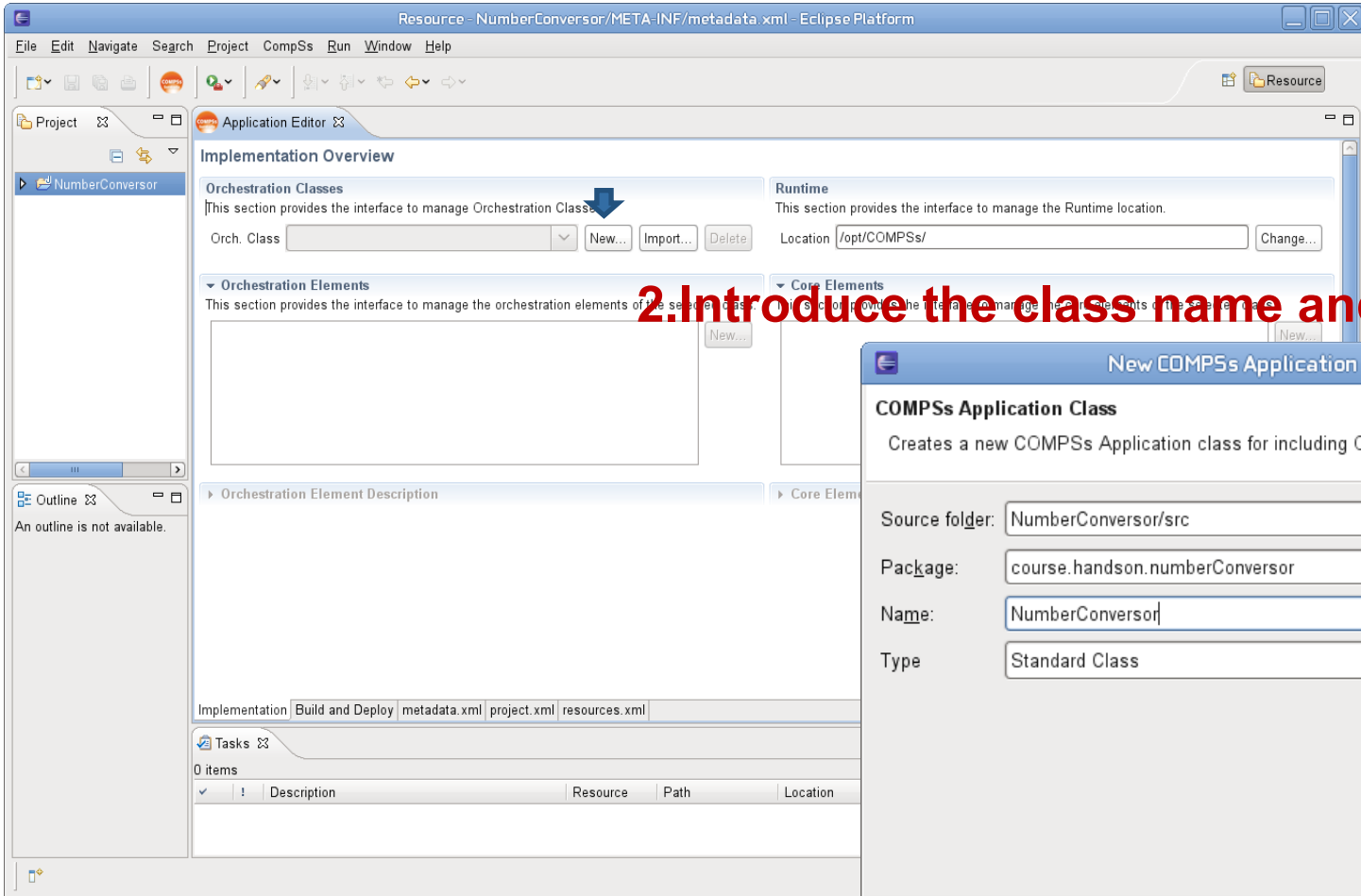
## 3. Introduce Project Details



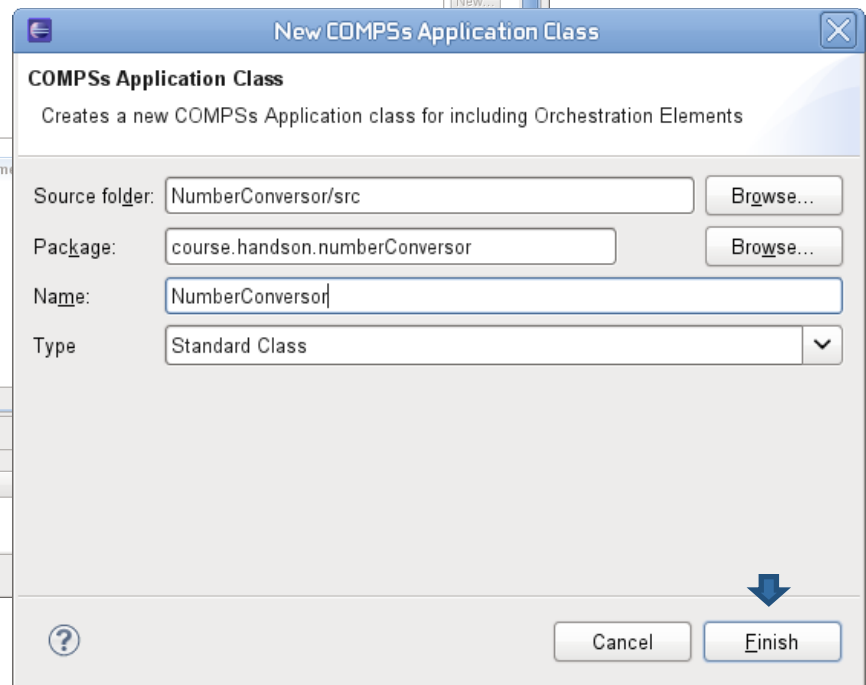
(Also available *CompSs ->Implementation->Create Application Project*)

# 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)



# 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**

The screenshot shows the Eclipse IDE interface. The 'Application Editor' is open, displaying the 'Orchestration Elements' section. A blue arrow points to the 'New...' button in this section. The 'New Orchestration Element' dialog box is open, showing the following configuration:

- Orchestration Element Location:
  - Source folder: NumberConverter/src
  - Package: course.handson.numberConverter
  - Orch. Class: NumberConverter
- Orchestration Element Description:
  - Name: main
  - Modifier:  private  static
  - Return Type: void
  - Parameters: 

Type	Name
String[]	args
  - Constraints: 

Name	Value
------	-------
- Part of the Service Interface

(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

The screenshot shows the Eclipse IDE interface. The 'New Core Element' dialog is open, with the 'New method core element from existing class method' option selected. The 'Core Elements' section of the Application Editor is visible, with a 'New...' button highlighted. Red arrows point from the 'New...' button to the 'New...' button in the dialog, and from the 'New...' button in the dialog to the 'Library Location' field in the 'Element description' section. Red text annotations provide instructions for selecting the jar file, class, and method.

**Library Location** `/home/user/ide_workspace/Conversor/jars/Conversor.jar` **← Select the jar file /home/user/ide\_workspace/Conversor/jars/Conversor.jar**

**Declaring Class** `Conversor` **← Select the Conversor class**

**Method** `convertToWords` **← 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

The image shows a sequence of Eclipse IDE windows illustrating the process of creating a core element:

- New Core Element (Top):** Shows the 'Create Core Element' dialog with 'New method core element from scratch' selected.
- New Core Element (Middle):** Shows the 'Define the Core Element' dialog with fields for Source folder, Package, CE Interface, Declaring Class (Merge), and Method (concatenate).
- New Core Element (Bottom):** Shows the 'New Core Element' dialog with 'Element description' fields for Name (concatenate), Return Type (String), and Parameters (origin, toAdd).
- Merge.java (Bottom):** Shows the source code for the Merge class with the concatenate method implemented.

```
package course.handson.numberConverter.coreelements;

public class Merge{

    public static String concatenate(String origin, String toAdd){
        return origin.concat(toAdd+" ");
    }
}
```

# IDE Hands On – Introduce the OE code

The screenshot shows the Eclipse IDE interface. The main editor displays the following Java code:

```
package course.handson.numberConverter;

import course.handson.converter.Converter;
import course.handson.numberConverter.coreelements.Merge;
import integratedtoolkit.types.annotations.Orchestration;

public class NumberConverter{

    @Orchestration
    public void main(String[] args){
        String concat = new String();
        for (String s:args){
            String word = Converter.convertToWords(s);
            concat = Merge.concatenate(concat, word);
        }
        System.out.println("Introduced numbers are: " + concat);
    }
}
```

A blue box highlights the `main` method, and a red arrow points to it with the text "Include the OE Code to call the CE methods".

**Include the OE  
Code to call the  
CE methods**

# IDE Hands On – Add conversor dependency to OE

The screenshot shows the IDE interface for 'NumberConversor.java'. The 'Orchestration Elements' section is active, showing a list of elements. A dialog box is open, allowing the user to add a new dependency. The dialog has a 'Type' dropdown set to 'JAR Library' and a 'Location' text field containing 'ars/Conversor.jar'. Below the dialog, the 'Orchestration Element Description' section is visible, with a table for dependencies and 'Add...' and 'Delete' buttons.

Type	Path
------	------

2.- Select the jar library:  
*/home/user/ide\_workspace/Conversor/  
jars/Conversor.jar*

1.- Click Add...



# IDE Hands On – Deploy Locally

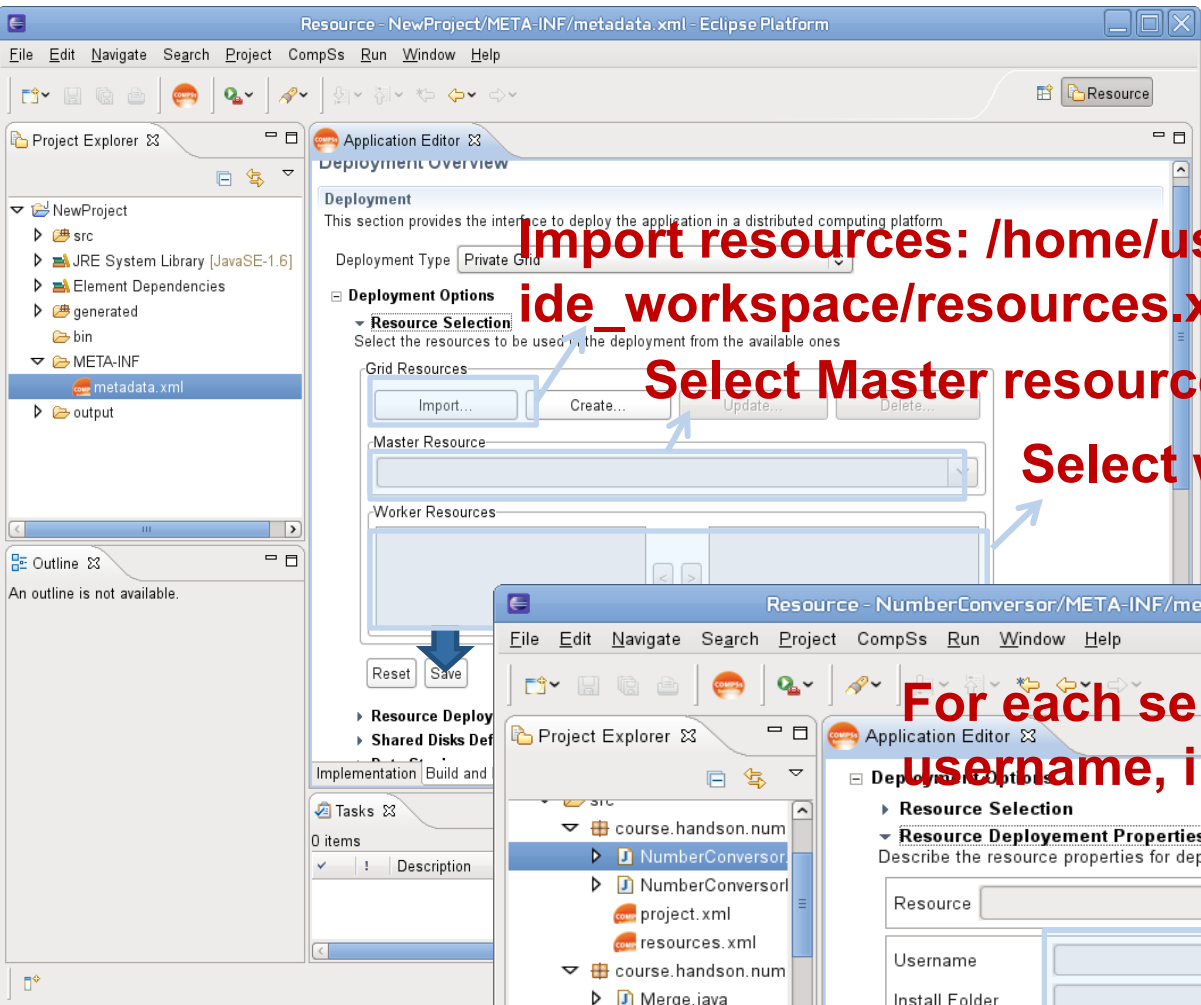
The screenshot shows the Eclipse IDE interface with the 'Deployment Overview' dialog open. The dialog is titled 'Deployment Overview' and contains the following elements:

- Deployment Type:** A dropdown menu set to 'Localhost'.
- Deployment Options:** A section containing two text input fields: 'App. Server Folder' and 'App. Elements Folder'. Both fields have a 'Select...' button to their right. A blue box highlights the 'App. Elements Folder' field and its 'Select...' button.
- Deploy Button:** A 'Deploy' button is located below the 'App. Elements Folder' field. A blue arrow points up to this button.

Below the dialog, a red text annotation reads: **Include location: /home/user/ide\_workspace/numConversor/**. A blue arrow points from this text to the 'App. Elements Folder' field in the dialog.

The background shows the Project Explorer on the left with a project named 'NumberConversor' selected. The Application Editor on the right shows the 'Deployment Overview' dialog. The Tasks view at the bottom is empty.

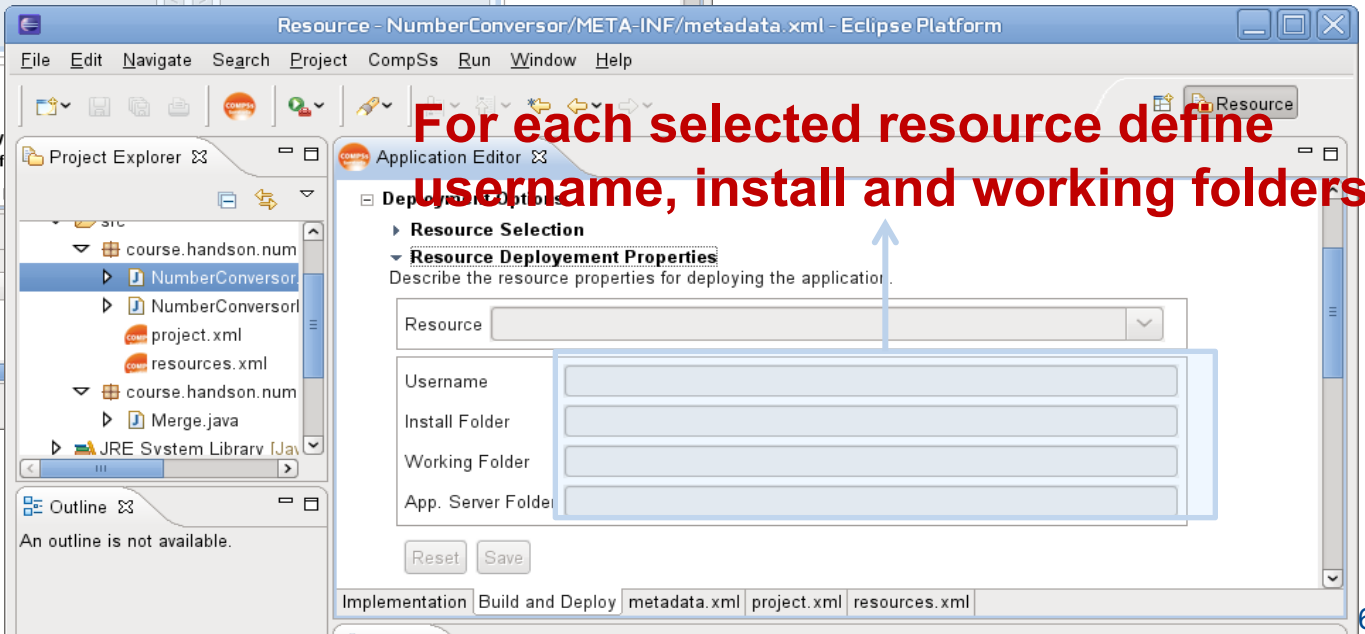
# IDE Hands On – Deploy Grid



**Import resources: /home/user/ide\_workspace/resources.xml**

**Select Master resource**

**Select worker resources**



**For each selected resource define username, install and working folders**

# 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

- Project page: <http://www.bsc.es/compss>
- Direct downloads page:  
<http://www.bsc.es/computer-sciences/grid-computing/comp-superscalar/download>
  - *Sample applications & development virtual appliances*
  - *Tutorials*
  - *Red-Hat & Debian based installation packages*
  - ...



[www.bsc.es](http://www.bsc.es)



**Barcelona  
Supercomputing  
Center**

*Centro Nacional de Supercomputación*

**Thank you!**

For further information please contact

[support-compss@bsc.es](mailto:support-compss@bsc.es)