Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

# Programming Distributed Computing Platforms with COMPSs

Workflows & Distributed Computing Group

31/01/2024

Barcelona & On-line

# Outline

## Day 2

- Session 6 (9:30-10:30): Programming Java Applications and Debug
- Session 7 (10:30-10:45): Cluster Hands-on Settings (MareNostrum4)
- **Coffee break (10:45 – 11:15)**
- Session 8 (11:15-13:00): Cluster Hands-on (MareNostrum4)
- **Lunch break (13:00 – 14:30)**
- Session 9 (14:30-15:30): Provenance with PyCOMPSs (with Hands-on)
- Session 10 (15:30-16:30): COMPSs with containers (with Hands-on)
- COMPSs Installation & Final Notes
- SLIDES
  - http://compss.bsc.es/releases/tutorials/tutorial-PATC_2024/

# Language differences overview

- Model and concepts are the same in all the Languages
- Differences in task declaration and synchronizations

| Language | Task declaration | Sychronization |
|---|---|---|
| **Python** | On method implementation | Explicit (compss_open, compss_wait_on, wait_on_file) |
| **Java** | Task Definition Interface | Implicit (except getFiles) |
| **C++** | Task Definition Interface | Explicit (compss_open, compss_wait_on, wait_on_file) |

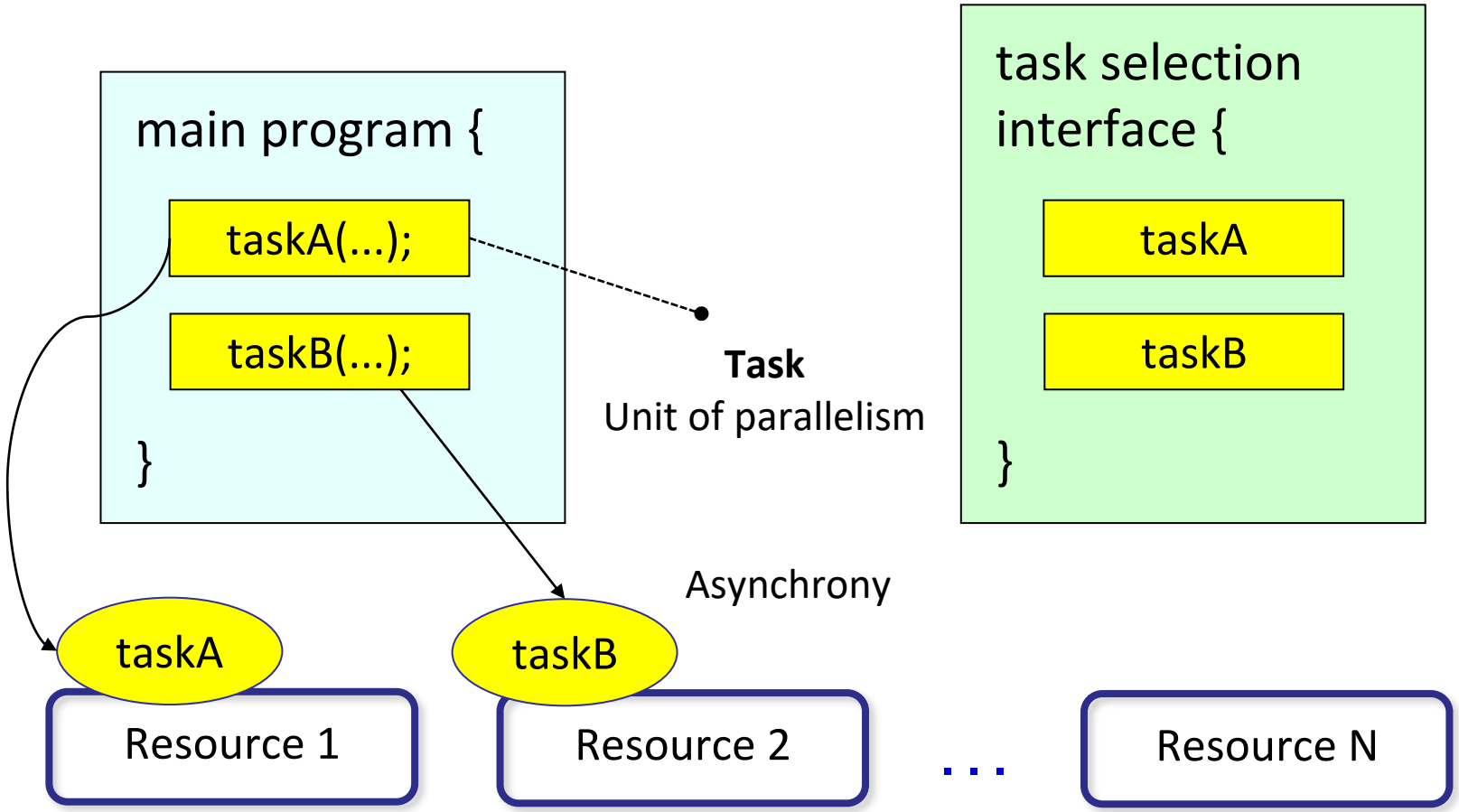
# Java Syntax


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Programming Steps

**1.** Identify tasks

**2.** Select tasks



# Task Selection Interface

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

   @HTTP(serviceName = "SampleService", resource = "sample/"),
         request = "POST", declaringClass = "sampleServiceImpl",
         payload = "{{query}}")

   Reply myServiceOp(
      @Parameter(name= "query", direction = IN) Query q
    );
}
```

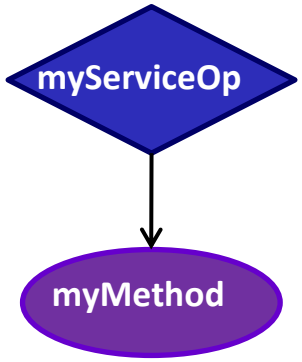
# Main program

```
public class App {

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

       Reply reply = myServiceOp(query);

       myMethod(reply, "out.txt");

       reply.printToLog();
   }
}
```

Service task call

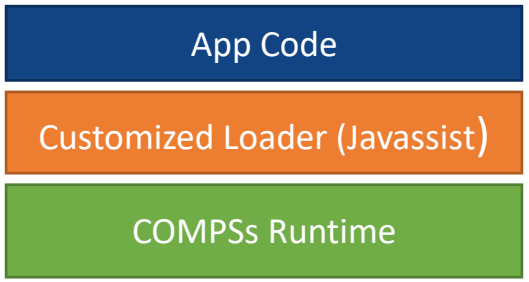
Method task call

Synchronization



# Why we do not need to synchronize?

- Code instrumented with Javassist
  - Modified at loading time.




```
public class App {
  public static void main(String[] args) {
      COMPSsRuntime.start();
      Query query = new Query(…);
      Reply reply = myServiceOp(query);   -> COMPSsRuntime.executeTask(…)
      myMethod(reply, "out.txt");    -> COMPSsRuntime.executeTask(…)
      COMPSsRuntime.getObject(reply);
      reply.printToLog();
      COMPSsRuntime.stop();
  }
}
```

# COMPSs API calls

- There are some calls that can not be inferred and the user can use calling the COMPSs API
  - Static class COMPSs
- Barrier: wait for all tasks to finish
  - COMPSs.**barrier**();
- Deregister object
  - As objects are registered in the runtime. It prevents the Java GC to delete the object.
  - COMPSs.**deregisterObject**(object);
- Synchronize a file without opening
  - COMPSs.**getFile**(filename);

# Java example


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Sample Application

- Main Program

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

   initializeCounters(counter1, counter2, counter3);

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

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

   }
}
```

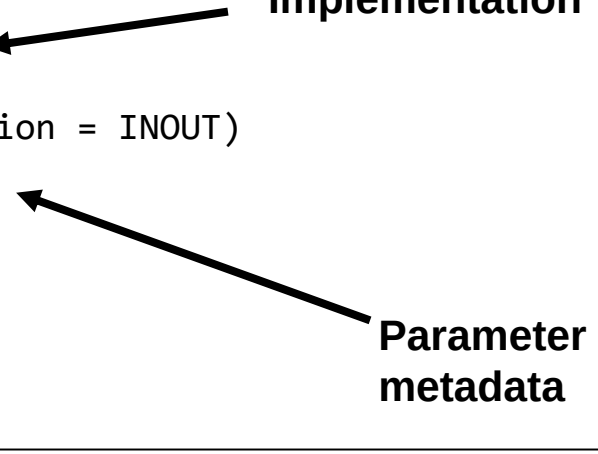
- Task Method

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

# Sample Application (Interface)

- Task Annotation Interface

```
public interface SimpleItf {

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

}
```



# Sample Application (Main Program)

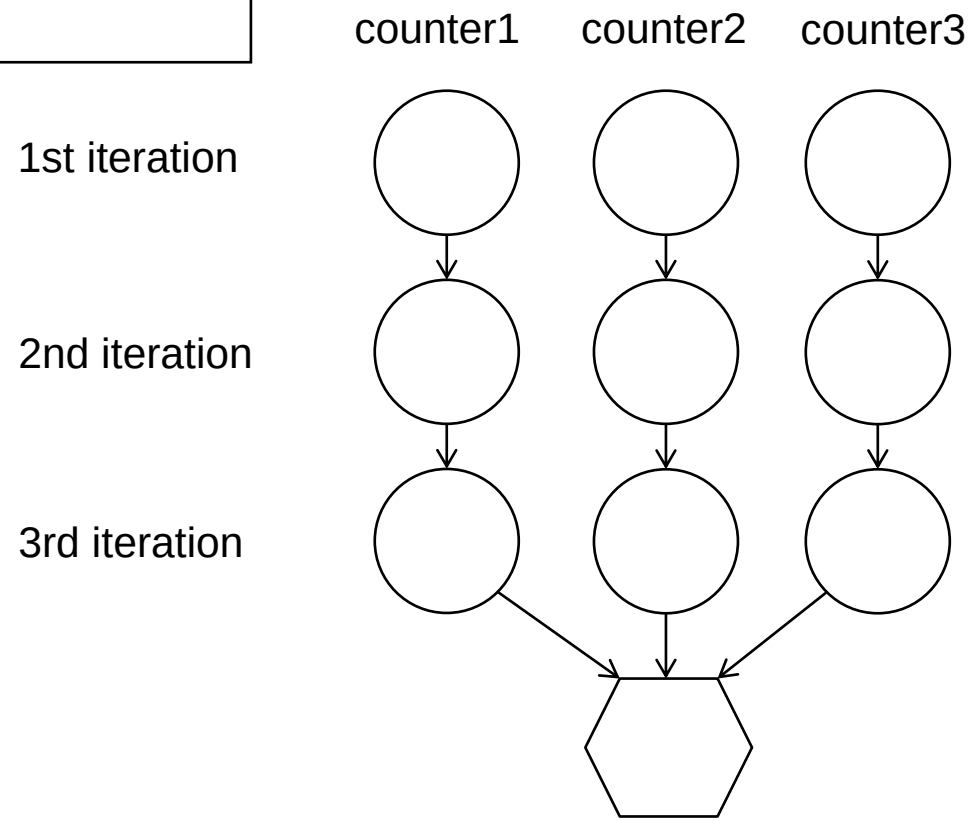
- Main program NO CHANGES!
- No need to synchronize data COMPSs is doing itself!

```
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);

   }
   printCounters(counter1, counter2, counter3);
}
```

No need to synch

# Sample Application: Task Graph

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



# Java Hands-on

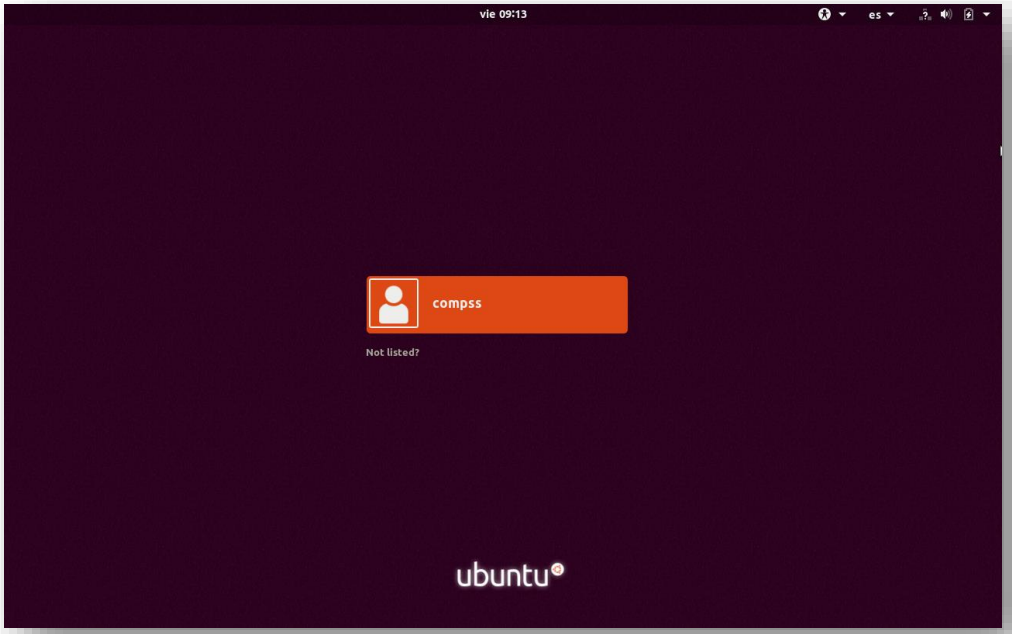

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Hands-on environment

- Windows users: Start the Virtual Machine
  - User: **compss**
  - Password: **compss2021**

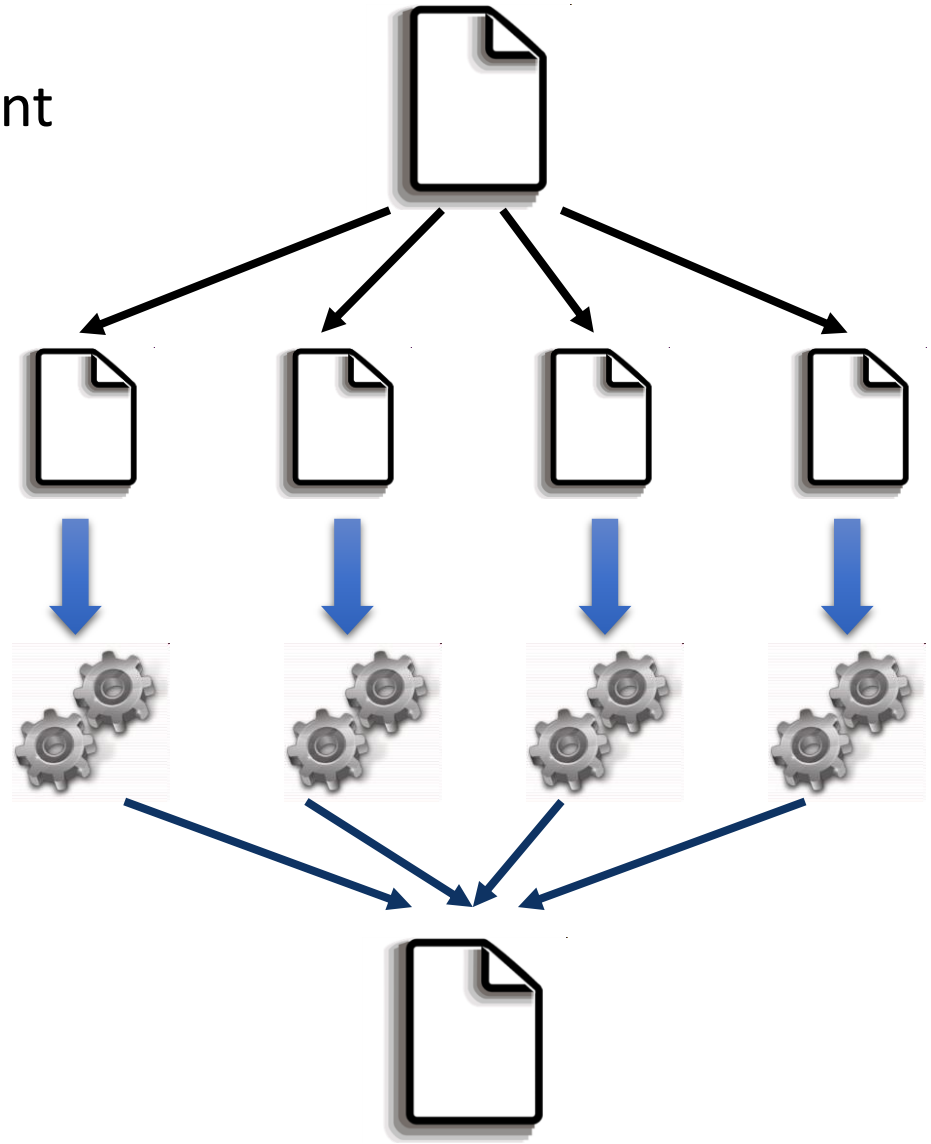
~/tutorial_apps/java

- Open IDE (eclipse, netbeans,..)



# Word count

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



# Java Hands On: Exercise

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

# Compilation and Simple Execution

- Compilation
  - Run ***mvn clean install*** in */home/compss/tutorial_apps/java/wordcount*

```
$ compss@bsc:~/> cd /home/compss/tutorial_apps/java/wordcount
$ compss@bsc:/home/compss/tutorial_apps/java/wordcount/> mvn clean package
```

- Init the docker testing environment

```
$ compss@bsc:/home/compss/tutorial_apps/java/wordcount/> compss init -n docker-tutorial docker \
      -i compss/compss-tutorial:3.3

$ compss@bsc:/home/compss/tutorial_apps/java/wordcount/> compss env change docker-tutorial
```

- Use compss command to run the application
  - **compss run** [options] < FQDN app. classname> <application args>
- ***Exercise:*** Simple word count execution
  - Usage:

    wordcount.uniqueFile.Wordcount <data_file> <block_size>


```
$ compss@bsc:/home/compss/tutorial_apps/java/wordcount/> compss run --classpath=jar/wordcount.jar \
      wordcount.uniqueFile.Wordcount data-set/file_small.txt 650
```

# Java Hands On: Exercise Solution

- Main Code

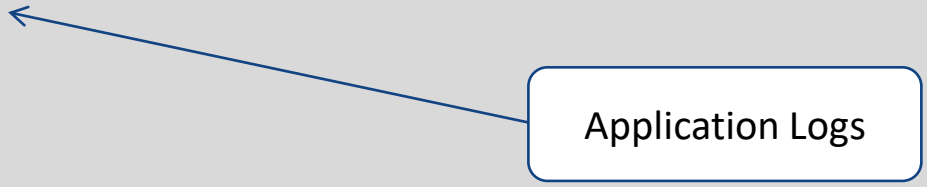
```
private static void computeWordCount() {
    HashMap<String, Integer> result  = new HashMap<String, Integer>();
    int start = 0;
    for (int i = 0; i < NUM_BLOCKS; ++i) {
      HashMap<String, Integer> partialResult = wordCountBlock(DATA_FILE, start, BLOCK_SIZE);
      start = start + BLOCK_SIZE;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

- Interface

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

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

# Java Hands-on: Result

```
$compss@bsc:~/tutorial_apps/java/wordcount/jar/> compss run –classpath=jar/wordcount.jar
    wordcount.uniqueFile.Wordcount data-set/file_small.txt 650

Executing  cmd: runcompss –-project=/project.xml --resources=/resources.xml …

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

WARNING: COMPSs Properties file is null. Setting default values

[   API]  -  Starting COMPSs Runtime v2.10 (build xxxx)
DATA_FILE parameter value = data-set/file_small.txt
BLOCK_SIZE parameter value = 650
[LOG] Computing word count result
[LOG] Counted Words is : 247
[   API]  -  No more tasks for app 1
[   API]  -  Getting Result Files 1
[   API]  -  Execution Finished

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



# Java Hands-On: Monitoring

- The runtime of COMPSs provides real-time monitoring to follow the progress of the executions
  - Running tasks, resources usage, execution time per task, real-time execution graph, etc.
- Start monitor and open browser

```
$compss@bsc:/home/compss/tutorial_apps/java/wordcount/> compss monitor start
```

```
$compss@bsc:/home/compss/tutorial_apps/java/wordcount/> firefox http://localhost:8080/compss-monitor
```

- Activate monitoring
  - Setting a monitoring interval
    - **compss run --monitoring=<int>**
  - With a default monitoring interval
    - **compss run –m** (or) **compss run --monitoring**
- ***Exercise:*** run wordcount enabling monitoring

```
$compss@bsc:/home/compss/tutorial_apps/java/wordcount/> compss run -m --classpath=jar/wordcount.jar
    wordcount.uniqueFile.Wordcount data-set/file_long.txt 350000
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Java Hands-on: Graph generation

- To generate the graph of an application, it must be run with the monitor or graph flags activated
  - **compss run -m | –graph | –g**
- The graph will be stored in:
  - **$HOME/.COMPSs/<APP_NAME>_<EX#>/monitor/complete_graph.dot**
- To convert the graph to a PDF format:
  - **compss gengraph <dot_file>**
- **Exercise:** generate the graph for the wordcount application

```
$compss@bsc:~/tutorial_apps/java/wordcount> compss run -g --classpath=jar/wordcount.jar \
    wordcount.uniqueFile.Wordcount data-set/file_small.txt 650
    …

$compss@bsc:~/tutorial_apps/java/wordcount> compss gengraph \
     .COMPSs/wordcount.uniqueFile.Wordcount_03/monitor/complete_graph.dot

Output file: /root/.COMPSs/wordcount.uniqueFile.Wordcount_04/monitor/complete_graph.pdf

$compss@bsc:~/tutorial_apps/java/wordcount> evince \
     .COMPSs/wordcount.uniqueFile.Wordcount_03/monitor/complete_graph.pdf
```



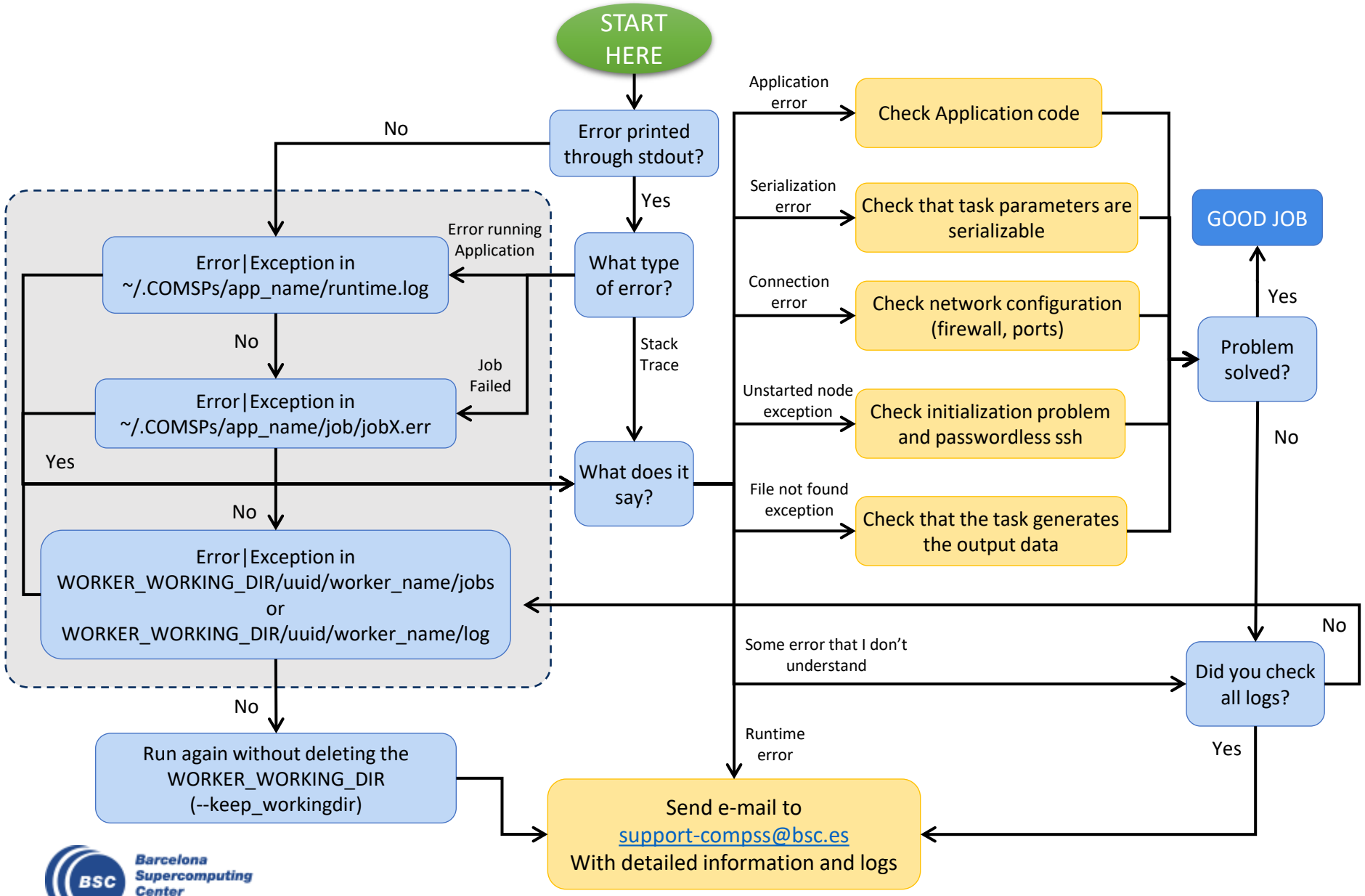
# Java Hands-on: Debugging

- Different log levels activated as options
  - **--log_level**=<level>
    (**off**: for performance | **info**: basic logging | **debug**: detect errors)
  - **--debug** or **-d**
- The output/errors of the main code of the application are shown in the console
- Logging files are stored by default in:
  - $HOME/.COMPSs/<APP_NAME>_XX
  - Customizable with flag **--log_dir=**<path>
- Inside this folder, the user can check :
  - The output/error of a task # N : ***jobs/jobN.[out|err]***
  - Messages from the COMPSs : ***runtime.log***
  - Worker: ***$HOME/.COMPSs/<app_name_XX>/workers***
    or during the execution at ***/<working_dir>/<uuid>/<node_name>/logs***
- ***Exercise:*** run wordcount with debugging

**Note:**
In **default environment,** the log folder is in ***$HOME/.COMPSs***.
In **Docker environments** executions, the **log_dir** has been set to **current directory**

```
$compss@bsc:/home/compss/tutorial_apps/java/wordcount/> compss run -d --classpath=jar/wordcount.jar
   wordcount.uniqueFile.Wordcount data-set/file_small.txt 650
```

# Debugging process




Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Demo

- Common errors:
  - Exceptions
    - In main code
    - Within a task
  - Usage of non-serializable objects
    - As a parameters
    - As a return
  - Connectivity problems
    - The master can not connect to the worker
    - The worker can not connect to the master


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