Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Programming Distributed Computing Platforms with COMPSs

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

Workflows & Distributed Computing Group

29-30/01/2019

Barcelona

Outline

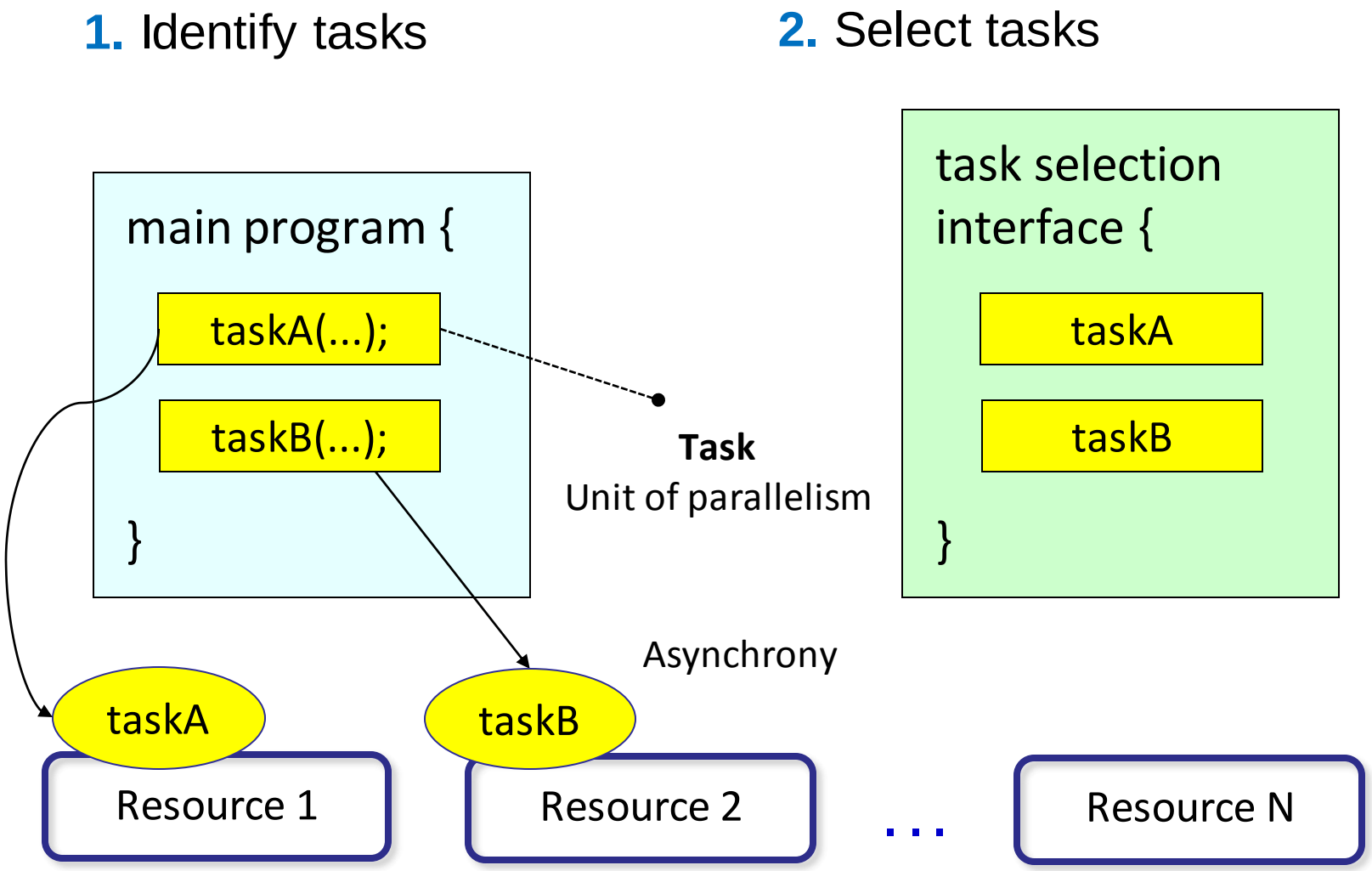
Day 2

- Session 6 (9:30-11:00): Java & C++
  - Writing Java applications
  - Java Hands-on
  - C++ Syntax
- Coffee break (11:00 – 11:30)
- Session 7 (11:30-12:30): COMPSs Advanced Features
  - Integration with OmpSs
  - Using binaries and MPI code
  - COMPSs execution environment
- Lunch break (13:30 – 14:30)
- Session 8 (14:30-15:30): Cluster Hands-on (MareNostrum)
- Session 9 (15:30 -16:30): Practical session (Bring your Own Code)
- COMPSs Installation & Final Notes
- SLIDES
  - http://compss.bsc.es/releases/tutorials/tutorial-PATC_2019/

Java Syntax

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Programming Steps



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
    );

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

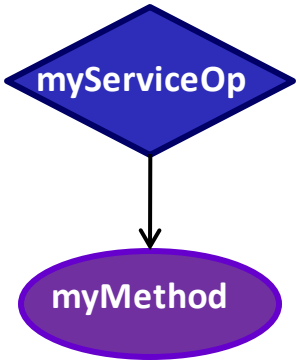
Main program

```
public class App {

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

       Reply reply = myServiceOp(query);

       myMethod(reply, "out.txt");

       reply.printToLog();
   }
}
```

Service task call

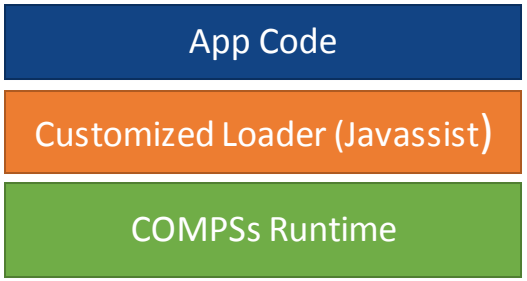
Method task call

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();
  }
}
```

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);

   }
}
```

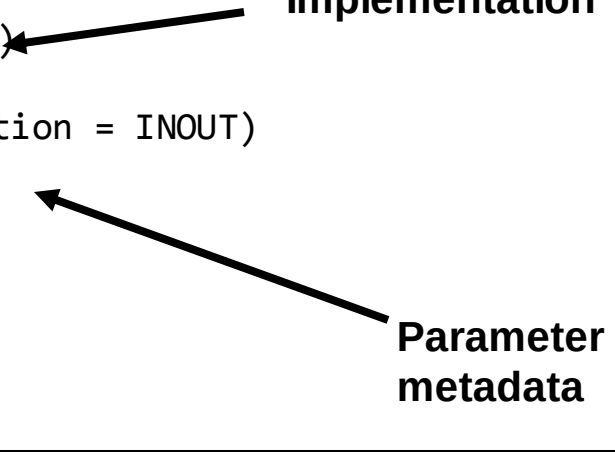
- Task Method

```
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)

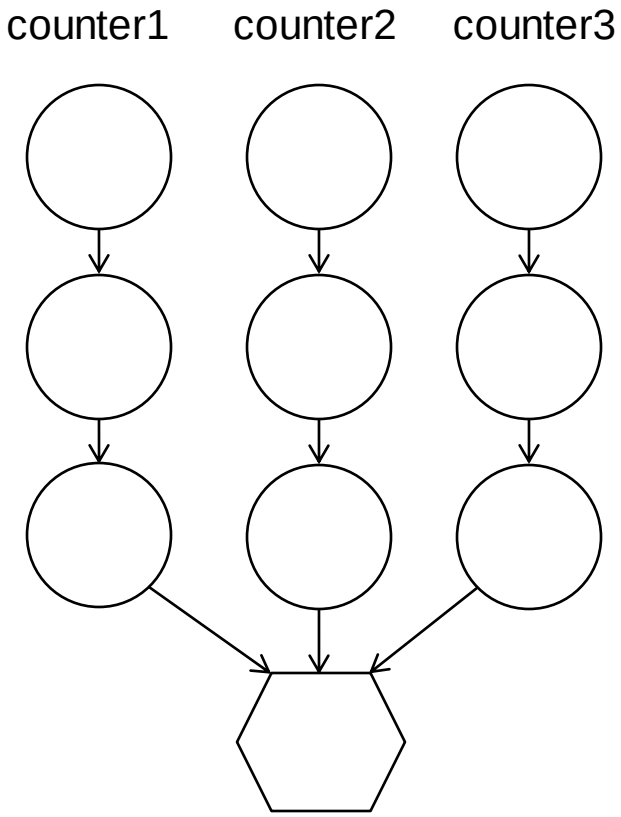
- Main program NO CHANGES!
- No need to synchronize data COMPSs is doing itself!

```
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

Programming Model: Task Graph

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



Other features

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Versioning

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

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);

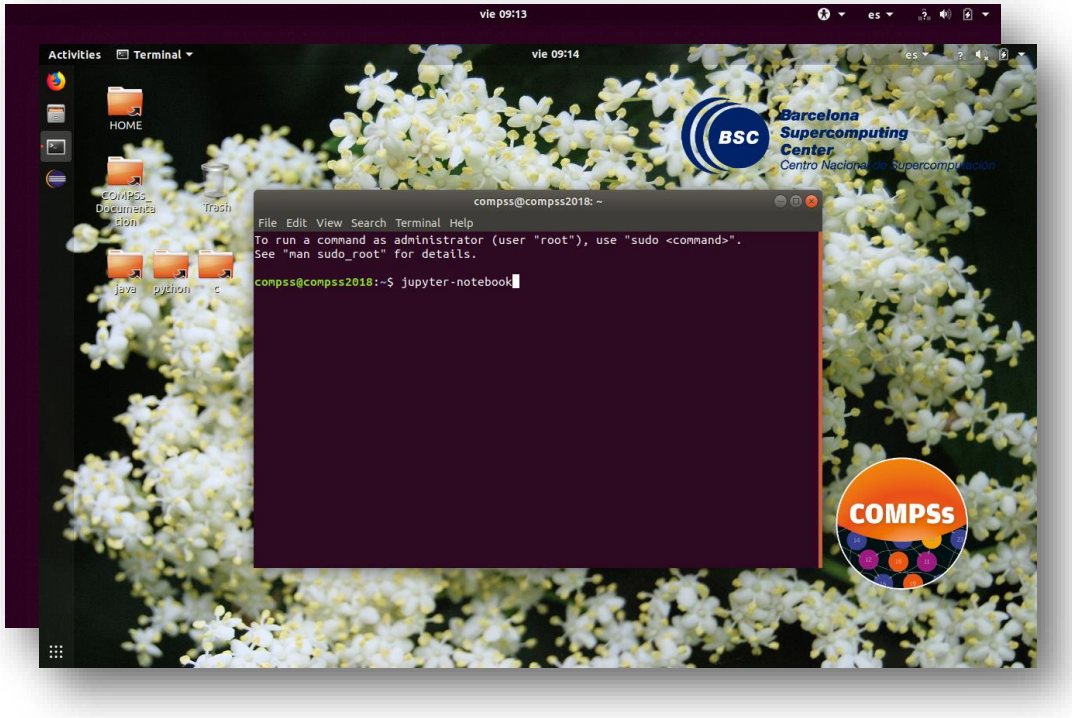
Java Hands-on

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

PyCOMPSs and Jupyter-Notebook in VM

- Start the Virtual Machine
  - User: **compss**
  - Password: **compss2018**
- Open eclipse

~/tutorial_apps/java



COMPSs in Docker

- In your machine
  - Install Eclipse IDE.
    - You can download from: https://www.eclipse.org/downloads/
  - Open Eclipse and install Maven for Eclipse plugin
  - Import maven tutorial_apps/java projects
- Run the compss-tutorial container (as root)

  $ *docker run –name mycompss -p 8080:8080 *
  *–v /path/to/tutorial_apps/:/home/compss/tutorial_apps *
  -itd compss/compss-tutorial:patc2019
  - NOTE: if docker daemon is not running: *sudo service docker start*
- Log into the container (as root)

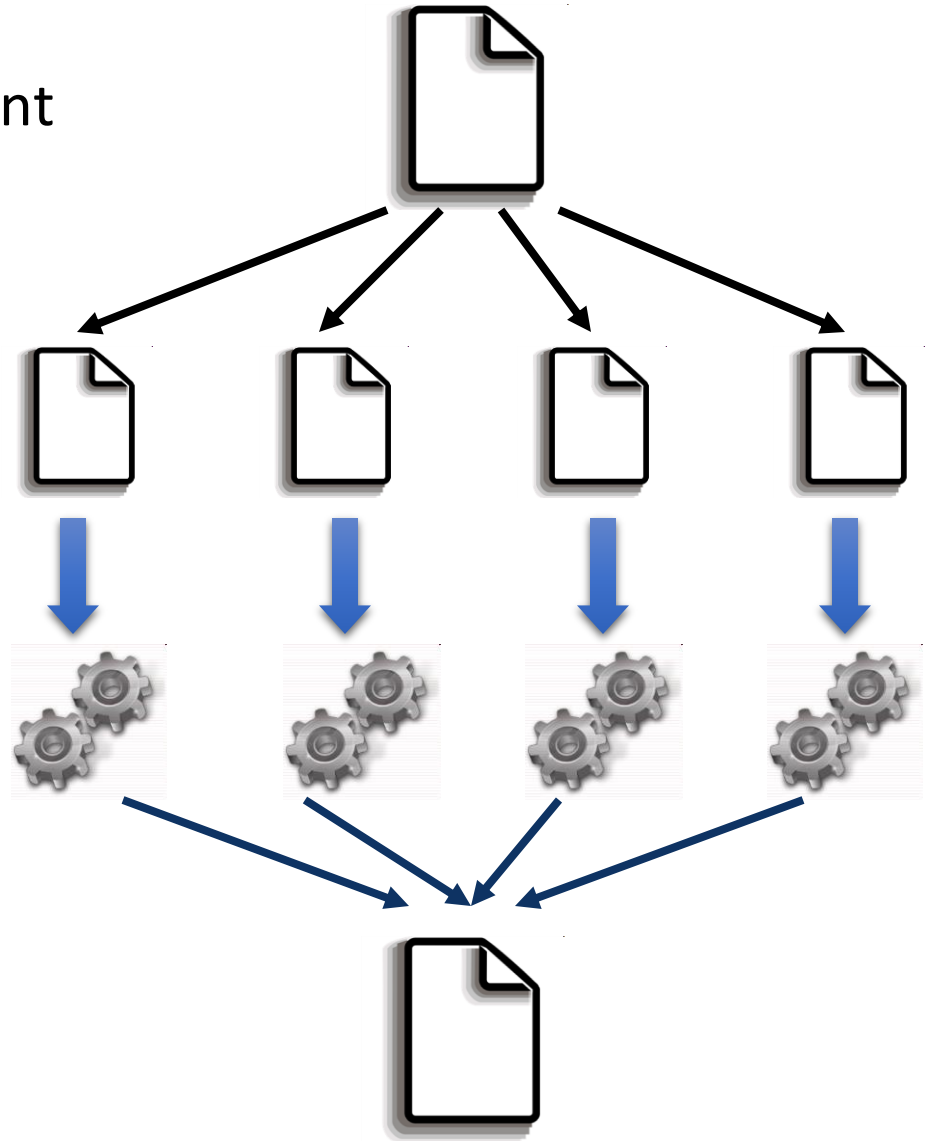
  $ *docker exec -it mycompss /bin/bash*

Data files

- Download & Copy data files to tutorial_apps folder
  - *cd /path/to/tutorial_apps/java/wordcount/data*
  - wget http://compss.bsc.es/releases/tutorials/tutorial-PATC_2019/data/file_short.txt
  - wget http://compss.bsc.es/releases/tutorials/tutorial-PATC_2019/data/file_long.txt

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*
- Use runcompss command to run the application
  - runcompss [options] < FQDN app. classname> <application args>

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

    wordcount.uniqueFile.Wordcount <data_file> <block_size>


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

Java Hands On: Exercise Solution

- Main Code

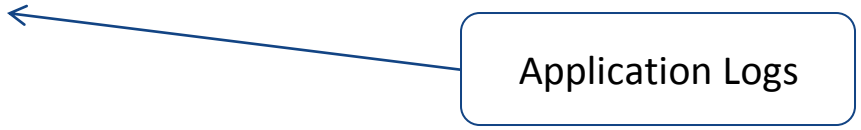
```
private static void computeWordCount() {
        HashMap<String, Integer> result  = new HashMap<String, Integer>();
        int start = 0;
        for (int i = 0; i < NUM_BLOCKS; ++i) {
          HashMap<String, Integer> partialResult = wordCountBlock(DATA_FILE, start, BLOCK_SIZE);
          start = 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/> runcompss wordcount.uniqueFile.Wordcount
    ~/tutorial_apps/java/wordcount/data/file_short.txt 650

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

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

WARNING: IT Properties file is null. Setting default values
[   API]  -  Deploying COMPSs Runtime v2.4 (build xxxx)
[   API]  -  Starting COMPSs Runtime v2.4 (build xxxx)
DATA_FILE parameter value = /home/compss/tutorial_apps/java/wordcount/data/file_short.txt
BLOCK_SIZE parameter value = 650
[LOG] Computing word count result
[LOG] Counted Words is : 250
[   API]  -  No more tasks for app 1
[   API]  -  Getting Result Files 1
[   API]  -  Execution Finished

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



Java Hands-on: Configuration

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

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

  - Other optional parameters
    - User, AppDir, LibraryPath

Java Hands-On: Configuration

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

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

Affects to application parallelism

Java Hands-On: Monitoring

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


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

Java Hands-on: Debugging

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


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

Java Hands-on: Graph generation

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


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

… application execution …

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

C Syntax

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

COMPSs C++ Binding

- Application Structure
- C Binding API
- Task definition /Supported data
- Compilation & Execution

Application Structure

- Main Code
  - <AppName>.cc
- Task definition interface
  - <AppName>.idl
- Task Implementation
  - <AppName>-functions.cc
- Auxiliary classes and methods
  - src folder

C++-Binding API

- Similar to the Python Binding
- Start/stop
  - compss_on() /compss_off()
- Synchronize and delete Objects
  - template <class T> void compss_wait_on(T* &obj);
  - template <class T> T compss_wait_on(T &obj);
  - template <class T> int compss_delete_object(T* &obj);
- Synchronize and delete files
  - void compss_ifstream(char * filename, ifstream& ifs);
  - void compss_ofstream(char * filename, ofstream& ofs);
  - void compss_delete_file(char * filename);
  - FILE* compss_fopen(char * filename, char * mode);
- Barrier
  - void compss_barrier();

Task definition

- Task definition interface (IDL-like interface)
  - Task definition
    - [return_type|void] [static][class_name::]method_name(params…);
  - Param definition
    - [in|out|inout] data_type name
  - Supported Data types for Dependencies
    - Files: *file*, objects: *Class_name*, 1D Arrays: *type[#elems]*
    - Primitive data types only IN direction: char*(string), int, double, float,…

```
interface example {
  @Constraints(ComputingUnits=2)
  void method(in f_in, out file f_out);
   //Expected C++ method: void method(char* f_in, char* f_out)

  ObjectEx ObjectEx:objectMethod(in inout ObjectEx accum,);
  //Expected C++ method: ObjectEx* ObjectEx:objectMethod(ObjectEx* accum)

  double[20] normal_method(in int n, in double[n] in_array);
  //Expected C++ method: double* normal_method(int n, double*)
}
```

Compilation & Execution

- Compilation
  - Generate master/worker stubs (C++ do not support reflection)
  - Command:
    - compss_build_app <appName>
- Execution
  - Same as Python/Java (runcompss command)
    - runcompss master/<appName> [app_args]
  - Require to set the AppDir
    - In the project.xml
    - Homogeneous cluster:
      - runcompss –appdir

```
<Project>
  <MasterNode/>
  <ComputeNode Name="localhost">
     <InstallDir>/opt/COMPSs/</InstallDir>
     <WorkingDir>/tmp/WorkerLocalhost/</WorkingDir>
     <Application>
        <AppDir>/home/tutorial_apps/c/matmul_objects/</AppDir>
     </Application>
  </ComputeNode>
</Project>
```

Exercise

- Matrix Multiplication with c-binding
  - <tutorial_apps>/c/matmul_object
- Compile:
  - compss_build_app Matmul
- Execute:
  - Edit xml/project.xml to set appdir
  - Runcompss –project=xml/project.xml master/Matmul <N> <M> <Value>


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