Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Programming Distributed Computing Platforms with COMPSs

Workflows & Distributed Computing Group

24-25/01/2023

Barcelona

COMPSs Advanced Features

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Outline

- Integrating external applications
  - Use of external Binaries, MPI, MPMD, @software
  - Failure management
  - Leveraging Numba
- Execution Environments

Other decorators: linking with other programming models

- A task can be more than a sequential function
  - A task in PyCOMPSs can be sequential, multicore or multi-node
  - External binary invocation: wrapper function generated automatically
  - Supports for alternative programming models: MPI and OmpSs
- Additional decorators:
  - @binary(binary="app.bin")
  - @mpi(binary="mpiApp.bin", runner="mpirun", processes=8)
  - @ompss(binary="ompssApp.bin")
- Can be combined with the @constraint and @implement decorators

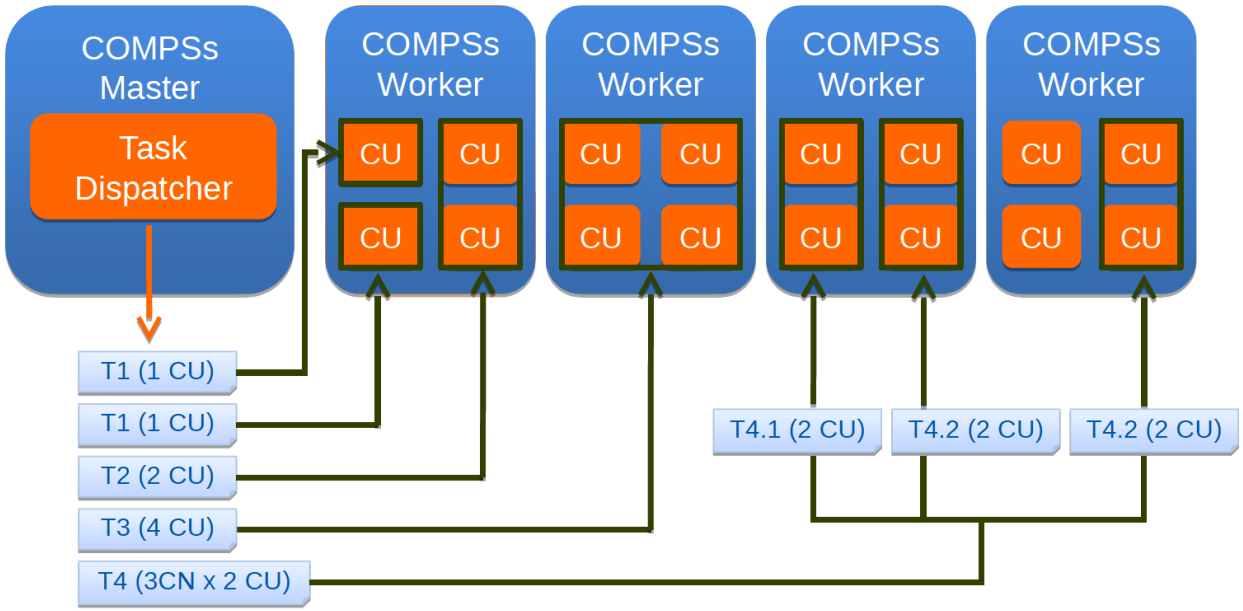
```
@binary(binary="app.bin", workingDir="/myApp")
@task()
def func(l):
    pass
```

Support for MPI tasks

- Extension of the PyCOMPSs interface
- Resource manager aware of multi-node tasks

```
@constraint (computingUnits= "2")
@mpi (runner="mpirun", processes= "3",...)
@task (returns=int, stdOutFile=FILE_OUT_STDOUT, stdErrFile=FILE_OUT_STDERR)
def nems(stdOutFile, stdErrFile):
    pass
```

Launches execution in
3 processes
2 threads / node



MPMD applications

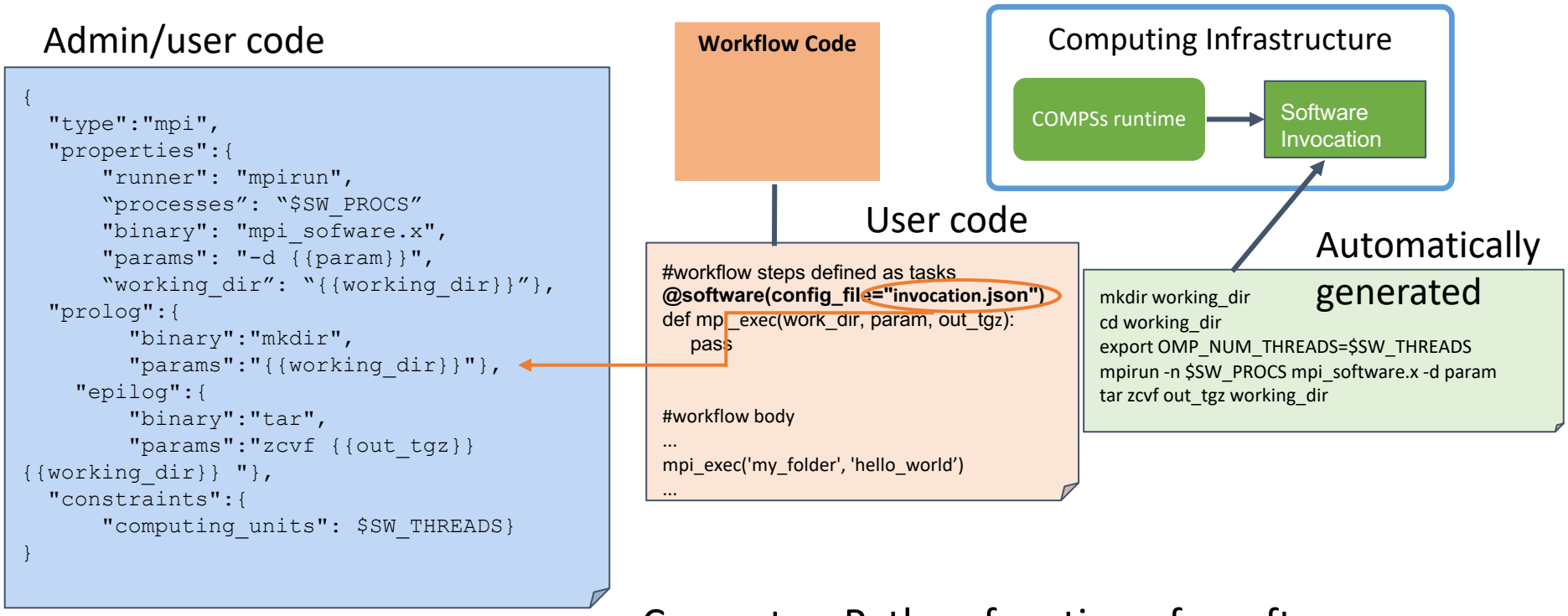
- The @mpmd_mpi decorator can be used to define Multiple Program Multiple Data (MPMD) MPI tasks

```
@mpmd_mpi(runner="mpirun", working_dir = {{working_dir_exe}},
          programs=[{binary="fesom.x", processes = "$FESOM_PROCS" },
                    {binary="oifs", args="-v ecmwf -e awi3", processes = "$OIFS_PROCS" },
                    {binary="rnfma", processes = "$RNFMA_PROCS"}])
@task(log_file={Type:FILE_OUT, StdIOStream:STDOUT}, working_dir_exe=DIRECTORY_INOUT)
def esm_simulation(log_file, working_dir_exe):
   pass
```

- As a result of the @mpmd_mpi annotation, the following commands will be generated:

```
> cd working_dir_exe; mpirun -n $FESOM_PROCS fesom.x : \
      -n $OIFS_PROCS oifs -v ecmwf -e awi3 : -n $RNFMA_PROCS rnfma
```

Software Invocation description



- Converts a Python function of a software invocation to a PyCOMPSs task
- Takes information from the description in the json file
- Enables reuse in multiple workflows

Hands-on

- Example of @binary
  - 7_Binary.ipynb

Failure management

- Interface than enables the programmer to give hints about failure management

```
@task(file_path=FILE_INOUT, on_failure='CANCEL_SUCCESSORS')
def task(file_path):
    ...
    if cond :
        raise Exception()
```

- Options: RETRY, CANCEL_SUCCESSORS, FAIL, IGNORE
- Implications on file management:
  - I.e, on IGNORE, output files: are generated empty
- **Possibility of ignoring part of the execution of the workflow, for example if a task fails in an unstable device**
- **Opens the possibility of dynamic workflow behaviour depending on the actual outcome of the tasks**

Hands-on

- Example of failure management
  - 3.3_Defining_classes_and_objects-with-fault-tolerance.ipynb

Task groups and exception management

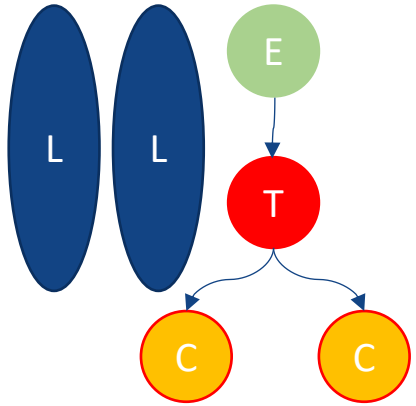
- Tasks can raise exceptions

```
@task(file_path=FILE_INOUT)
def comp_task(file_path):
    ...
    raise COMPSsException("Exception raised")
```

- Combined with groups of tasks enables to cancel the group of tasks on the occurrence of an exception

```
def test_cancellation(file_name):
    try:
        with TaskGroup('failedGroup'):
            long_task(file_name)
            long_task(file_name)
            executed_task(file_name)
            comp_task(file_name)
            cancelledTask(FILE_NAME);
            cancelledTask(FILE_NAME)

except COMPSsException:
        print("COMPSsException caught")
        write_two(file_name)
```



Leveraging NUMBA

- Just in time compilation

```
@task(returns=1)
def ident_loops(x):
    r = np.empty_like(x)
    n = len(x)
    for i in range(n):
        r[i] = np.cos(x[i]) ** 2 + np.sin(x[i]) ** 2
    return r
```

```
@task(returns=1, numba=True)
def ident_loops_jit(x):
    r = np.empty_like(x)
    n = len(x)
    for i in range(n):
        r[i] = np.cos(x[i]) ** 2 + np.sin(x[i]) ** 2
    return r
```

Leveraging NUMBA

- Exemple of use
  - 8_Integration_with_Numba

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Integrating Binaries (Java)

```
public interface SampleItf {
  @Binary(binary = "/path/to/binary")
  void binaryTask(
    @Parameter(type = Type.STRING, direction = Direction.IN) String message,
    @Parameter(type = Type.FILE, direction = Direction.IN, prefix="–in=") String fileIn,
    @Parameter(type = Type.FILE, direction = Direction.OUT, prefix="–out") String fileOut,
    @Parameter(type = Type.FILE, direction = Direction.OUT, stream = Stream.STDERR) String
   fileErr
 );
 // command: /path/to/binary message –in=fileIn –out=fileOut 2>fErr
}
```

```
import binary.BINARY;
…
  public static void main(String[] args) {
    //Binary Task invocation
    BINARY.binaryTask("message", "fileIn", "fileOut", "fileErr");
    …
 }
```

```
package binary;
public class BINARY {
  public static void binaryTask( String message, String fileIn,
   String fileOut, String fileErr){
      /* Dummy implementation, just to compile*/
   }
}
```

Integrating MPI (Java)

```
public interface SampleItf {
  @MPI(binary = "/path/to/binary", mpiRunner = "mpirun", processes = "2")
  @Constraints(computingUnits = "2")
  void mpiTask(
    @Parameter(type = Type.STRING, direction = Direction.IN) String opt1,
    @Parameter(type = Type.FILE, direction = Direction.OUT, prefix="—out") String fileOut
  );
 // command: mpirun -np 4 -H node1,node1,node2,node2 /path/to/binary opt1 -out=fileOut
}
```

```
import binary.BINARY;
…
  public static void main(String[] args) {
    // MPI Task invocation
     MPI.mpiTask("option1", "fileOut");
     …
  }
```

```
package mpi;
public class MPI{
  public static void mpiTask(String opt1, String
    fOut){
    /* Dummy Implementation just to compile */
  }
}
```

Failure management – Java syntax

Interface

```
public interface TestTimeOutItf {
  @Method(declaringClass = "testTimeOut.TestTimeOutImpl", timeOut = "3000",
     onFailure = OnFailure.IGNORE)
  void timeOutTaskSlow(@Parameter(type = Type.FILE, direction = Direction.INOUT)
          String fileName);

  @Method(declaringClass = "testTimeOut.TestTimeOutImpl", timeOut = "3000")
  void timeOutTaskFast(@Parameter(type = Type.FILE, direction = Direction.INOUT)
          String fileName);
}
```

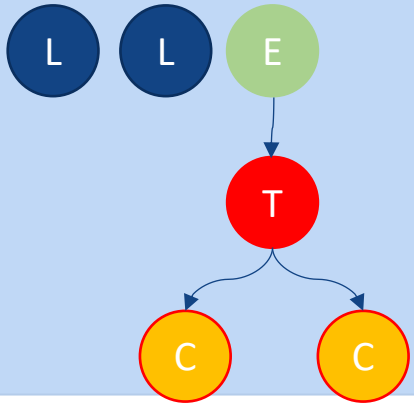
Task Implementation

```
public static void timeOutTaskSlow(String filename) throws Exception {
        try {
            Thread.sleep(5000);
            writeFile(filename, String.valueOf(6));
        } catch (InterruptedException e1) {
            e1.printStackTrace();
        }
        System.out.println("Before cancellation point");
        // Cancellation point to check time out
        COMPSsWorker.cancellationPoint();
        System.out.println("After the cancellation point");
    }
```

Task groups and exception management - Java

```
public static void throwException(String fileName) throws Exception {
        System.out.println("Exception is going to be thrown");
        throw new COMPSsException("Second task threw an exception");
}
```

```
private static void testCancelation() throws InterruptedException {
        try (COMPSsGroup a = new COMPSsGroup("FailedGroup", true)) {
            System.out.println("Executing task group that throws COMPSsException ");
            // Long tasks that will be cancelled while being executed
            CancelRunningTasksImpl.longTask(FILE_NAME);
            CancelRunningTasksImpl.longTask(FILE_NAME);
            // Short task correctly executed
            CancelRunningTasksImpl.executedTask(FILE_NAME);
            // The exception is thrown by the second task of the group
            CancelRunningTasksImpl.throwException(FILE_NAME);
            // These two tasks are cancelled before being executed
            CancelRunningTasksImpl.cancelledTask(FILE_NAME);
            CancelRunningTasksImpl.cancelledTask(FILE_NAME);
        } catch (COMPSsException e) {
            // CancelRunningTasksImpl.writeTwo(FILE_NAME);
            System.out.println("Exception caught!!");
        } catch (Exception e1) {
            e1.printStackTrace();
        }
        for (int j = 0; j < N; j++) {
            CancelRunningTasksImpl.writeTwo(FILE_NAME);
        }
    }
```




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