Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Orchestration of Software Packages in Data Science Workflows

Cristián Ramón-Cortés
Javier Conejero
Jorge Ejarque
Rosa M. Badia

May 2019

Outline

- **Introduction**
  - Motivation
  - COMPSs / PyCOMPSs
  - Current annotations (Python only)
- **Integration**
  - New annotations
  - Exit value, prefix and I/O Stream annotations
- **Use Case: NMMB-MONARCH**
  - NMMB-MONARCH
  - Parallelization design
  - Evaluation
- **Conclusions and Future Work**

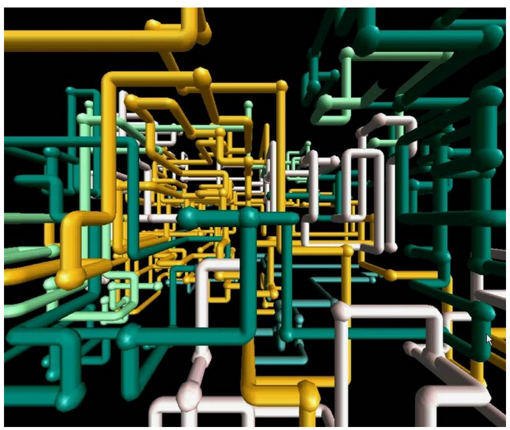
Introduction

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Motivation

- Data Science applications:
  - Complex pipelines developed by field experts
  - Widely used state of the art software packages for specific actions
  - Heterogeneous requirements

Cumbersome handmade pipelines



Programmability
Performance
Scalability

Specialized Frameworks

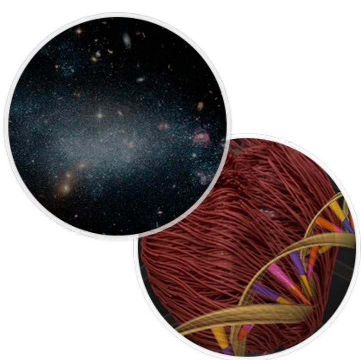
Kepler, DASK, FireWorks, COMPSs, Galaxy, Parsl, slurm workload manager, nextflow, AUTOSUBMIT, swift

COMPSs Motivation

Parallel Issues
- Identifying parallel regions
- Concurrency management
- Execution orchestration

Distributed Issues
- Remote execution
- Data transfers

Ease the development of distributed applications



THE GOAL:

Any field expert can scale up an application to thousands of cores

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

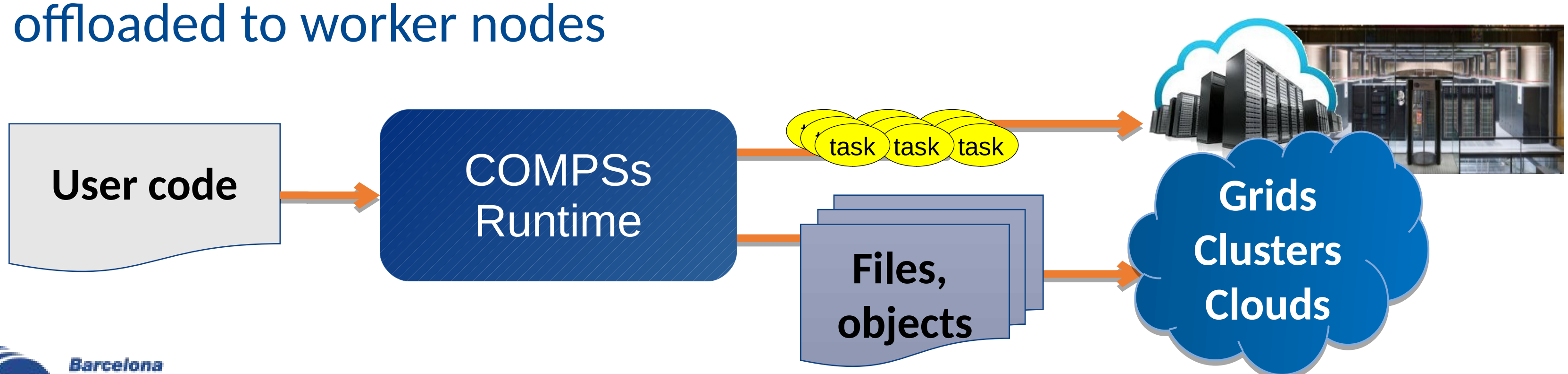
COMPSs

- Based on sequential programming
- Minimal impact on user code
  - General purpose programming language + annotations

Java + Annotated Interface   python + @decorator

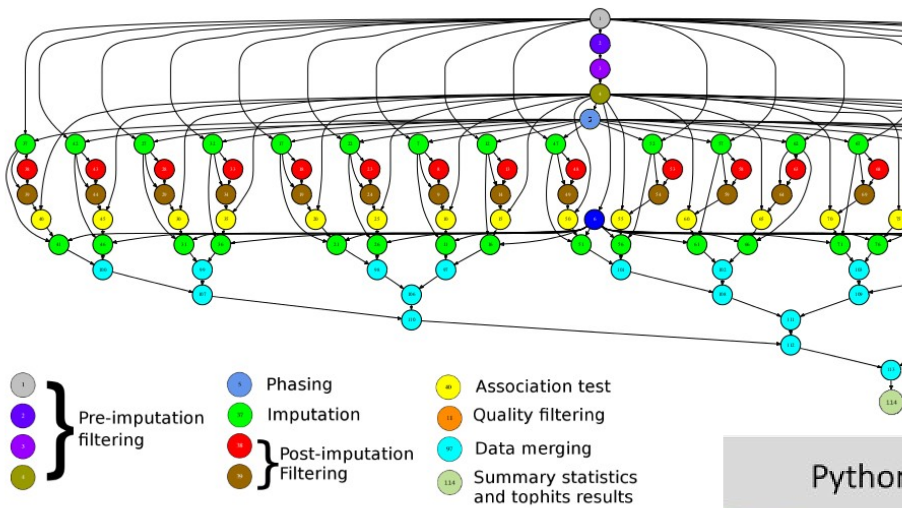
- Aimed at exploiting the inherent parallelism of sequential applications on distributed environments.

- Sequential execution starts in the master node, and tasks are offloaded to worker nodes

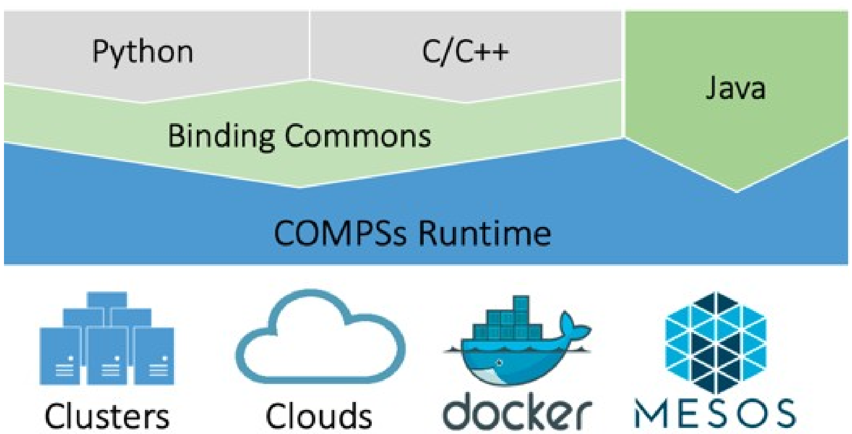


COMPSs

- Task-based programming model
  - Task is the unit of work
  - Implicit Workflow: Builds a task graph at runtime that expresses potential concurrency



- Infrastructure agnostic
  - Same application runs on clusters, grids, clouds, and containers



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

PyCOMPSs Annotations

- Python decorators for task selection + synchronization API
  - Instance and class methods
  - Task data directions

```
@task(a=IN, b=IN, c=INOUT)
def multiply_acum(a, b, c):
    c += a * b
```

```
@task(returns=int)
def multiply(a, b, c):
    return c + a * b
```

```
@constraint(computingUnits="2")
@task(file=FILE_IN)
def my_task(x):
    ...
```

```
@binary(binary="sed")
@task(f=FILE_INOUT)
def binary_task(flag, expr, f):
    pass
```

```
@task(returns=dict)
def wordcount(block):
  ...

@task(result=INOUT)
def reduce(result, pres):
  ...

def main(a, b, c):
  for block in data:
    pres = wordcount(block)
    reduce(result, pres)
  result = compss_wait_on(result)

  # f = compss_open(fn)
  # compss_delete_file(f)
  # compss_delete_object(o)
  # compss_barrier()
```

Integration

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

New Programming Model annotations (1)

- **Binaries:** Execution for regular binaries (i.e., BASH, fortran, C)
  - Binary
  - Working Directory *(opt)*

```
@binary(binary = "path_to_bin")
@task()
def myBinaryTask():
  pass
```

- **OmpSs:** Execution of OmpSs binaries
  - Binary
  - Working Directory *(opt)*

```
@ompss(binary = "path_to_bin")
@task()
def myOmpSsTask():
  pass
```

- **MPI:** Execution of MPI binaries
  - Binary
  - MPI Runner
  - Computing Nodes
  - Working Directory *(opt)*

```
@mpi(mpi_runner = "mpirun",
     binary = "path_to_bin",
     computing_nodes = "N")
@task()
def myMPITask():
  pass
```

New Programming Model annotations (2)

- **COMPSs:** Nested COMPSs applications
  - Application name
  - Runcompss command
  - Runcompss extra flags *(opt)*
  - Computing Nodes
  - Working Directory *(opt)*

```
@compss(runcompss = "runcompss",
        app_name = "mpirun",
        computing_nodes = "N")
@task()
def myNestedCOMPSsTask():
  pass
```

- **MultiNode:** Native Java/Python multi-node tasks
  - Computing Nodes

```
@multinode(computing_nodes = "N")
@task()
def myMultiNodeTask():
  # Python code
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

New Programming Model annotations (3)

Exit value

```
@binary(binary = "binary")
@task(returns=int)
def task_ev():
  pass
```

```
./binary; ev=$?
```

I/O Stream Parameters

```
@binary(binary = "binary")
@task(file_in=FILE_IN_STDIN,
      file_out=FILE_OUT_STDOUT | FILE_INOUT_STDOUT,
      file_err=FILE_OUT_STDERR | FILE_INOUT_STDERR)
def task_io(param, file_in, file_out, file_err):
  pass
```

```
./binary < file_in > file_out >&2 file_err
```

Parameters Prefix

```
@binary(binary = "binary")
@task(file1=FILE_IN,
      file2={Type: FILE_INOUT, Prefix: "-q="},
      k_value={Prefix: "k"})
def task_prefix(p="-p", file1=None, file2=None, k_value=10):
  pass
```

```
./binary -p file1.in -q=file2.inout k10
```

Use Case: NMMB-MONARCH

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

NMMB-Monarch

- **M**ultiscale **O**nline **N**onhydrostatic **A**tmosphe**R**e **Ch**emistry
  - Multiscale: Global to regional scales allowed (up to 1km)
  - Fully on-line coupling: weather-chemistry feedback processes allowed
  - Enhancement with data assimilation system

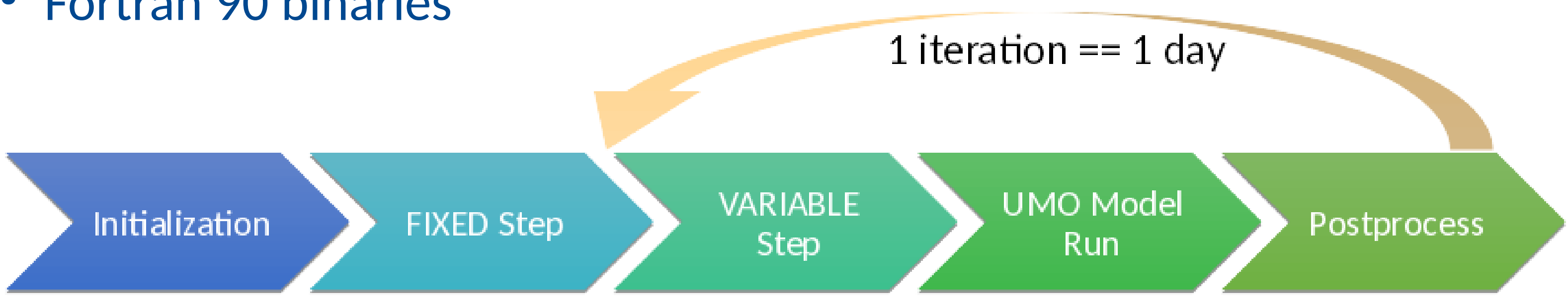
Objective:
Predict the atmospheric life cycle

- The model couples online the NMMB with the gas-phase and aerosol continuity equations to solve the atmospheric chemistry processs in detail

- Designed to account for the feedback among gases, aerosol particles, and meteorology

NMMB-Monarch

- Originally:
  - BASH workflow
  - Fortran 77 binaries
  - Fortran 90 binaries



Initialization
- Read configuration files
- Setup simulation parameters
- Prepare the environment (output folders)

FIXED Step
- Actions to setup the FIXED step of the execution
- Build the fixed executables
- Generate initial data (e.g. mountains, CO2, temperature, etc.)
- Cleanup

VARIABLE Step
- Compile more variable binaries
- Begin binary calls (e.g. Prepare climatological vegetation fraction, Prepare dust related variable, etc.)
- Cleanup

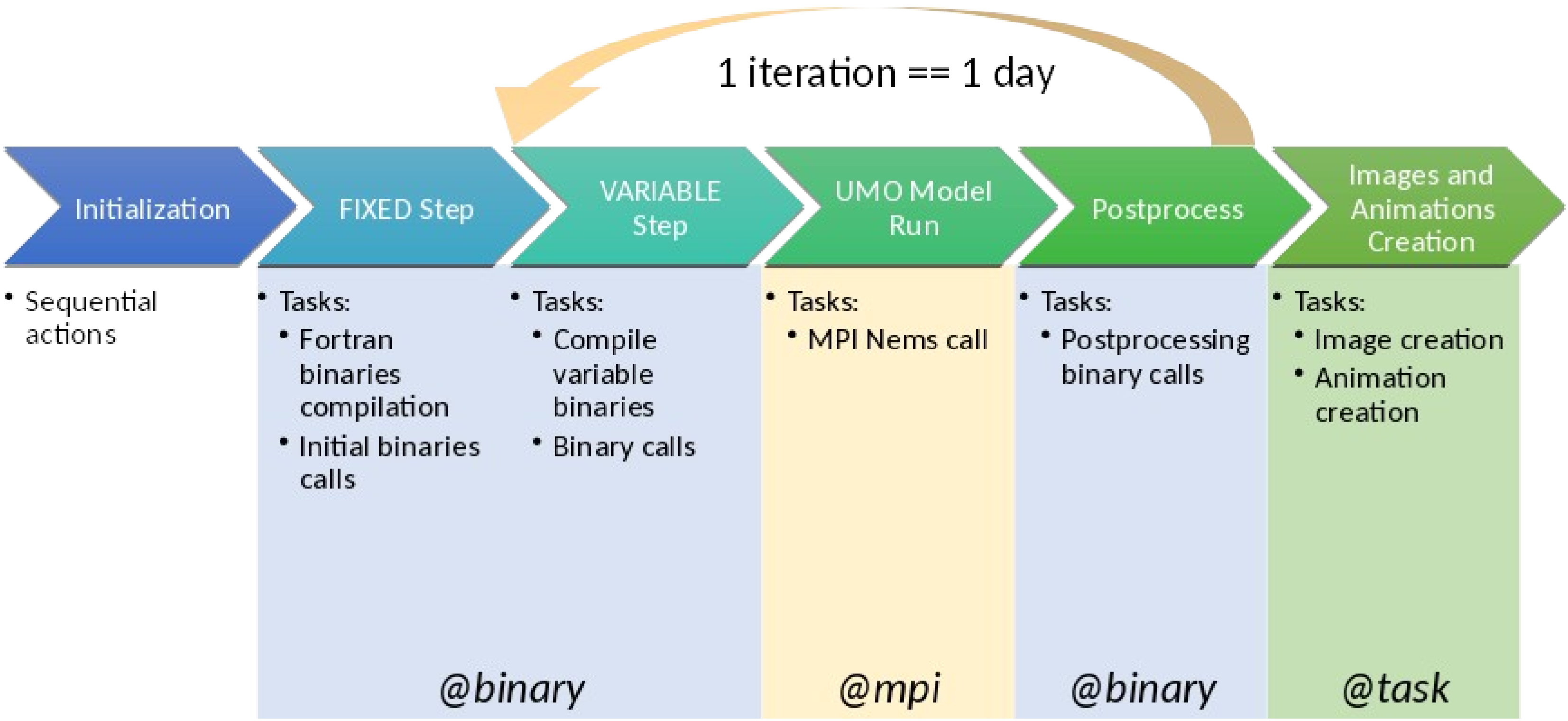
UMO Model Run
- Actions to prepare the UMO Model execution
- Perform the UMO Model simulation step (NEMS call)
- Actions to perform after the UMO Model execution

Postprocess
- Simulation post processing
- Generation of the final simulation result

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Parallelization with COMPSs/PyCOMPSs

- Migrate the workflow code to sequential Java / Python code keeping the same structure
- Determine the potential tasks
- Include the creation of images and animations



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Task Annotations

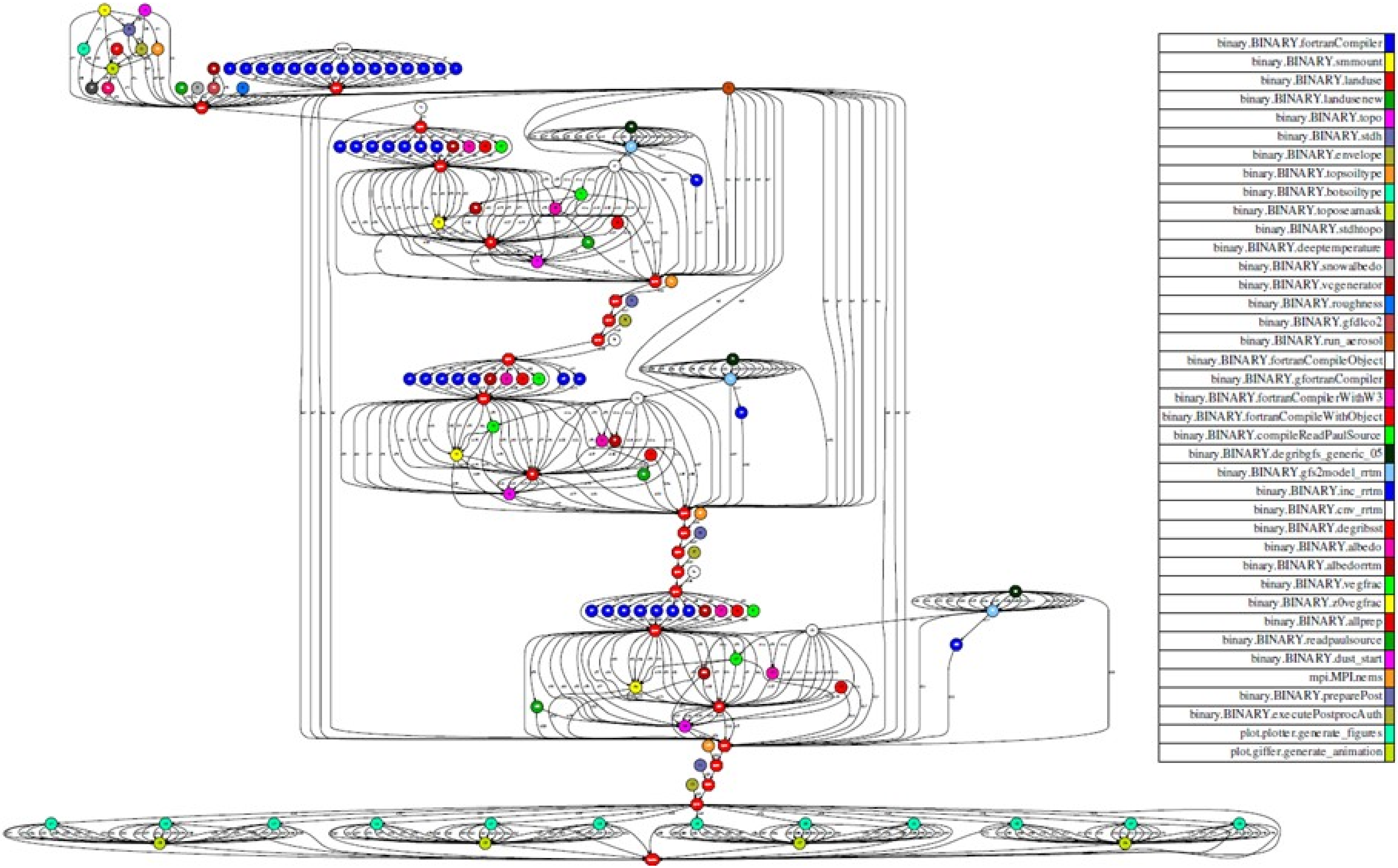
```
@binary(binary="deeptemperature.x")
@task(returns=int,
      seamask=FILE_IN,
      deep_temperature=FILE_OUT)
def deeptemperature(seamask,

deep_temperature):
  pass
```

```
@constraint(computingUnits="16")
@mpi(mpi_runner="mpirun",
     binary="/path/to/NEMS.x",
     computing_nodes="$NEMS_NODES",
     working_dir="/path/to/nems/out")
@task(returns=int,
      stdout_file=FILE_OUT_STDOUT,
      stderr_file=FILE_OUT_STDERR)
def nems(stdout_file, stderr_file):
  pass
```

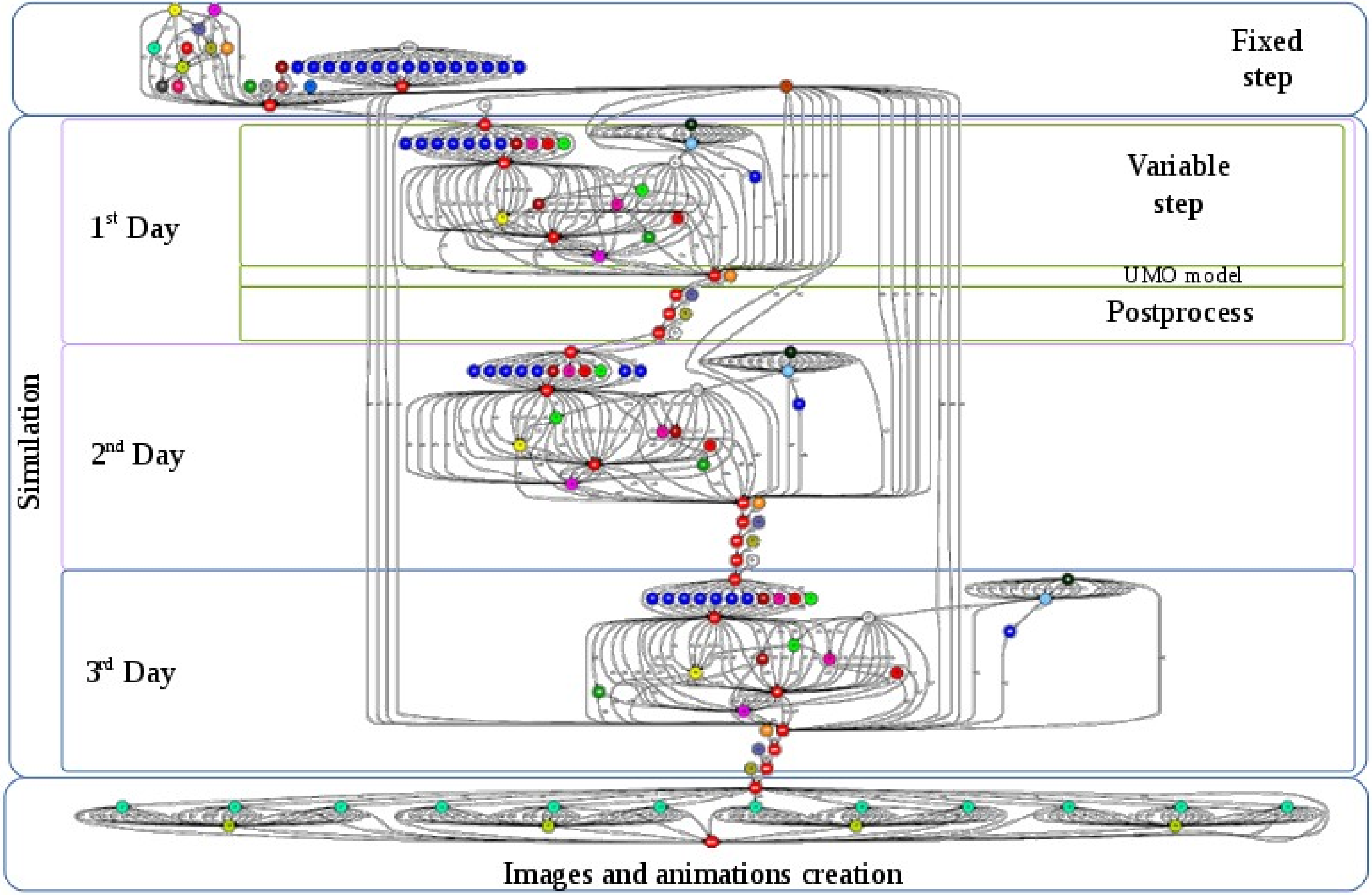
```
@task(fname=FILE_IN,
      i1=FILE_OUT, i2=FILE_OUT, i3=FILE_OUT, i4=FILE_OUT, i5=FILE_OUT,
      i6=FILE_OUT, i7=FILE_OUT, i8=FILE_OUT, i9=FILE_OUT)
def generate_figures(date, fname, vname, i1, i2, i3, i4, i5, i6, i7, i8, i9):
  # Python code
```

```
@task(gif_name=FILE_OUT, varargsType=FILE_IN)
def generate_figures(fig_name, skip_frames, *args):
  # Python code
```

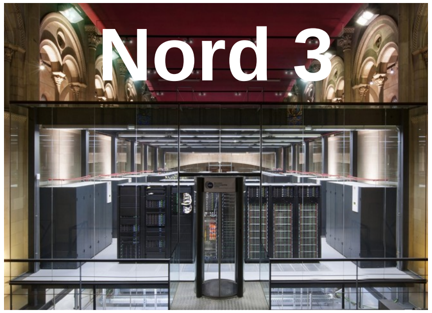
Task graph


binary.BINARY.fortranCompiler
binary.BINARY.smmount
binary.BINARY.landuse
binary.BINARY.landusenew
binary.BINARY.topo
binary.BINARY.stdh
binary.BINARY.envelope
binary.BINARY.topsoiltype
binary.BINARY.botsoiltype
binary.BINARY.toposeamask
binary.BINARY.stdhtopo
binary.BINARY.deeptemperature
binary.BINARY.snowalbedo
binary.BINARY.vcgenerator
binary.BINARY.roughness
binary.BINARY.gfdlco2
binary.BINARY.run_aerosol
binary.BINARY.fortranCompileObject
binary.BINARY.gfortranCompiler
binary.BINARY.fortranCompilerWithW3
binary.BINARY.fortranCompileWithObject
binary.BINARY.compileReadPaulSource
binary.BINARY.degribgfs_generic_05
binary.BINARY.gfs2model_rrtm
binary.BINARY.inc_rrtm
binary.BINARY.cnv_rrtm
binary.BINARY.degribsst
binary.BINARY.albedo
binary.BINARY.albedorrtm
binary.BINARY.vegfrac
binary.BINARY.z0vegfrac
binary.BINARY.allprep
binary.BINARY.readpaulsource
binary.BINARY.dust_start
mpi.MPI.nems
binary.BINARY.preparePost
binary.BINARY.executePostprocAuth
plot.plotter.generate_figures
plot.giffer.generate_animation

Task graph

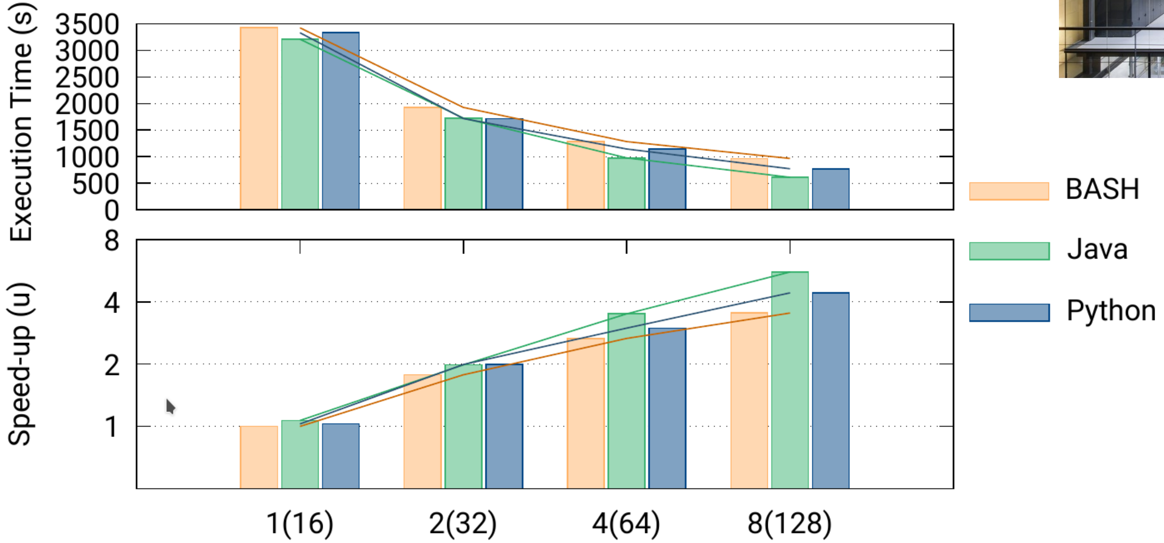


Barcelona Supercomputing Center
Centro Nacional de Supercomputación



Performance

- Strong Scaling. 3 Days simulation

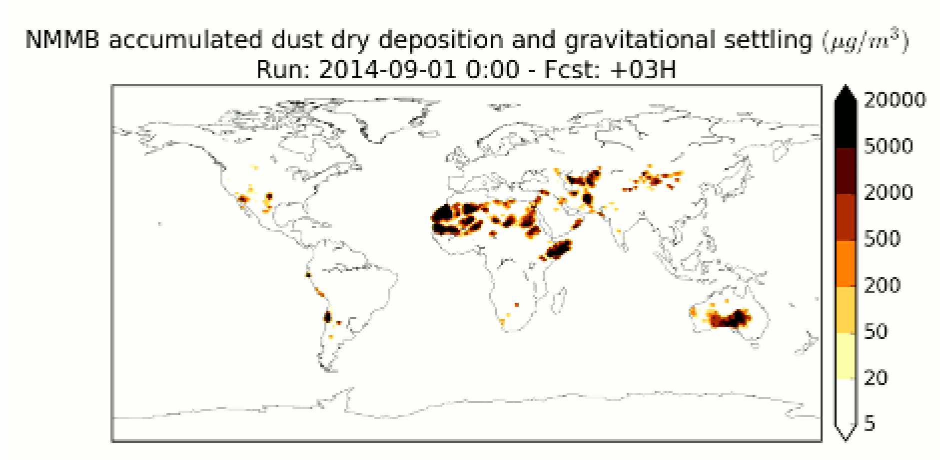


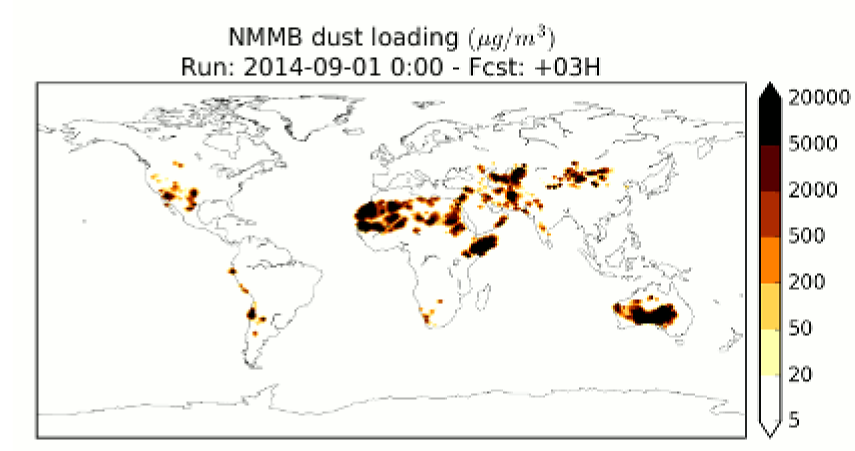
- Per step analysis. 1 Day simulation @ 4 workers (64 cores)

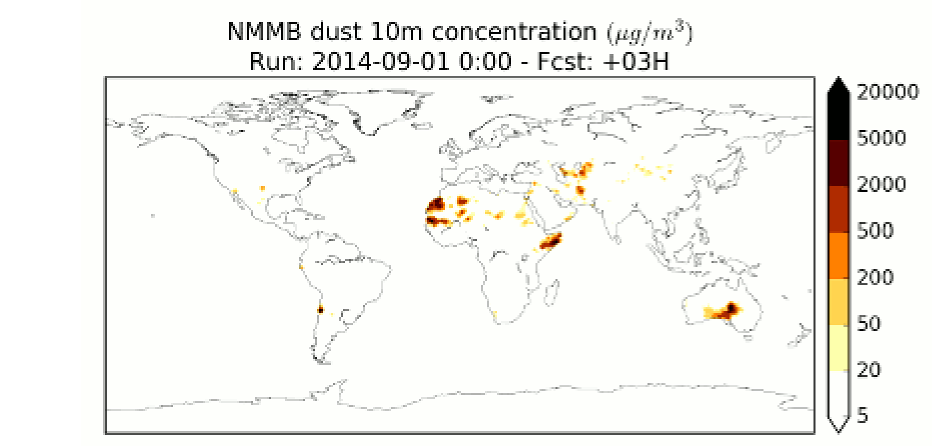
	Execution Time (s)			Speed-up (u)	
Step	**BASH**	**Java**	**Python**	**Java**	**Python**
Fixed	290	117	119	2.48	2.43
Variable	26	19	22	1.37	1.18
Model Sim.	244	242	239	1.01	1.02
Post Process	38	34	33	1.12	1.15
Total	**598**	**412**	**413**	**1.45**	**1.45**

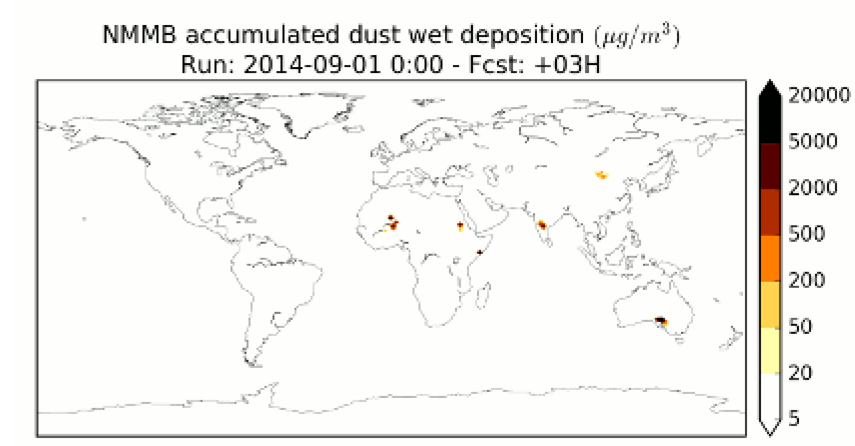
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Simulation Results









Conclusions and Future Work

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Conclusions and Future Work

- **Enabling the orchestration of Software Packages in Data Science workflows**
  - Complex workflows in a single language (Java or Python) with an homogeneous annotation for many software packages
  - Transparent orchestation, data management, and execution of binaries, OmpSs, MPI, nested COMPSs, and native multi-node tasks

- **NMMB-MONARCH has been parallelized with COMPSs and PyCOMPSs (Java and Python workflows)**
  - Task level parallelization with Binaries, MPI, and native functions
  - Programmability and performance improvements

- **Next steps**
  - Extend the annotation for more software packages
  - Pre/post actions when spawning non-native tasks

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Thank you

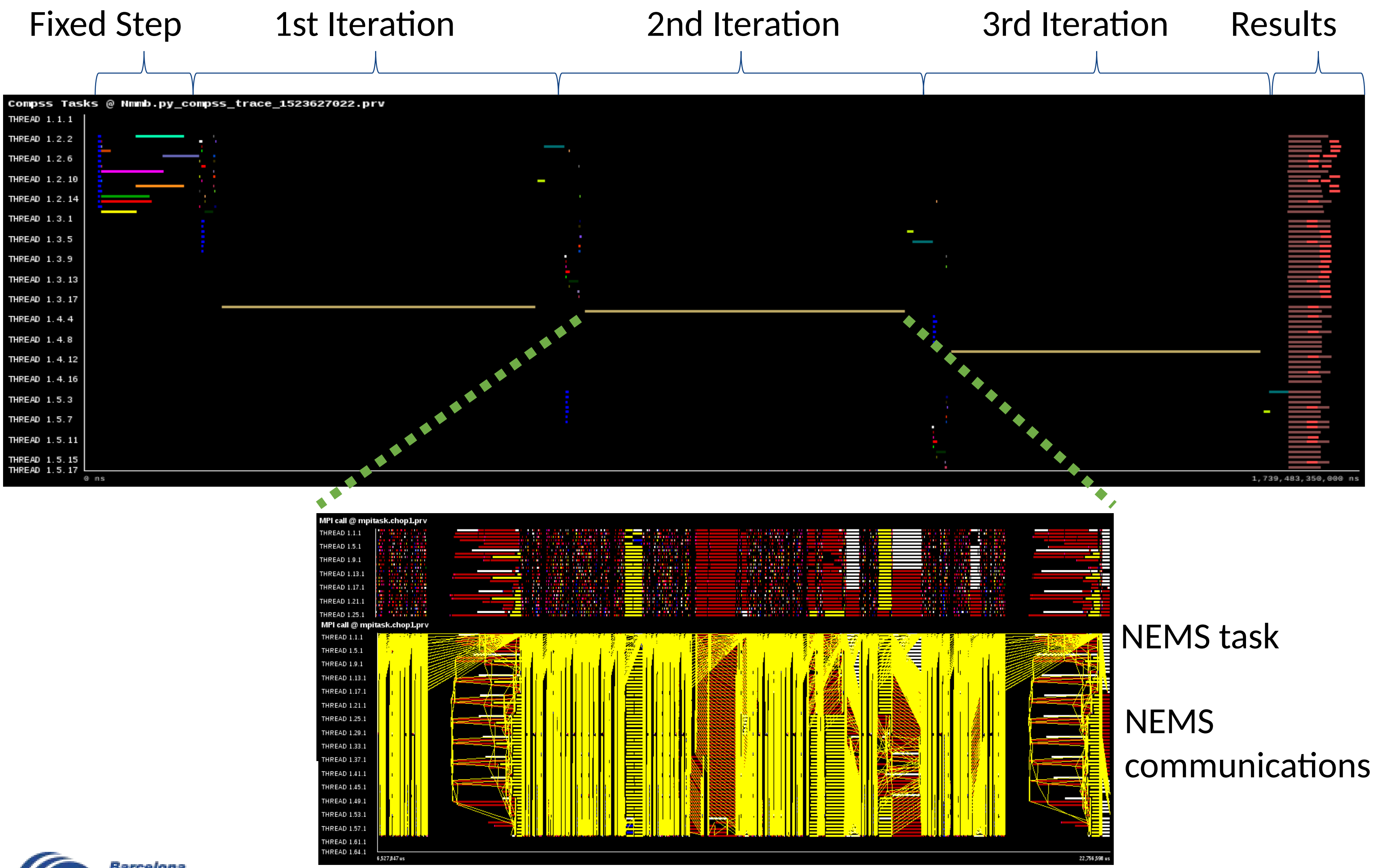
cristianrcv/pycompss-autoparallel

http://compss.bsc.es/

cristian.ramon-cortes@bsc.es

Backup

Application Behaviour



Programmability

- Better configuration management
- Better object-oriented structure
- Improves maintenance, extension, and debugging

Original NMMB-MONARCH Workflow				
Language	*Files*	*Blank*	*Comment*	*Code*
Fortran 90	23	394	2806	7581
Fortran 77	8	182	3568	6518
BASH	16	185	134	776

New NMMB-MONARCH Workflow with COMPSs/PyCOMPSs				
Language	*Files*	*Blank*	*Comment*	*Code*
Fortran 90	23	394	2806	7581
Fortran 77	8	182	3568	6518
Java	18	560	890	2721
Python	18	515	710	2399