Mathematical Modelling, Simulation and Optimization for Societal Challenges with Scientific Computing

HPC and Cloud Resources for Running Mathematical Simulations Efficiently

Javier Carnero, Víctor Sande, Carlos Fernández, F. Javier Nieto, IBERGRID 2018, Lisbon

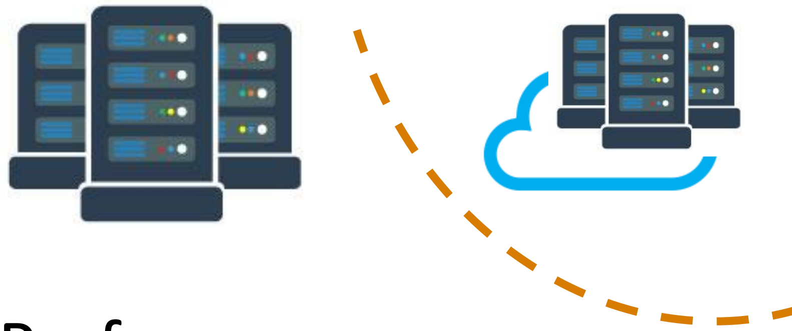
Outline

- Why HPC & Cloud?
- Proposed Solution
- Use Case
- Conclusions & Future Work

Why HPC & Cloud?

- + Flexibility
- + Provision
- + Ease of Usage
- + €€€

- - Performance
- - €€€

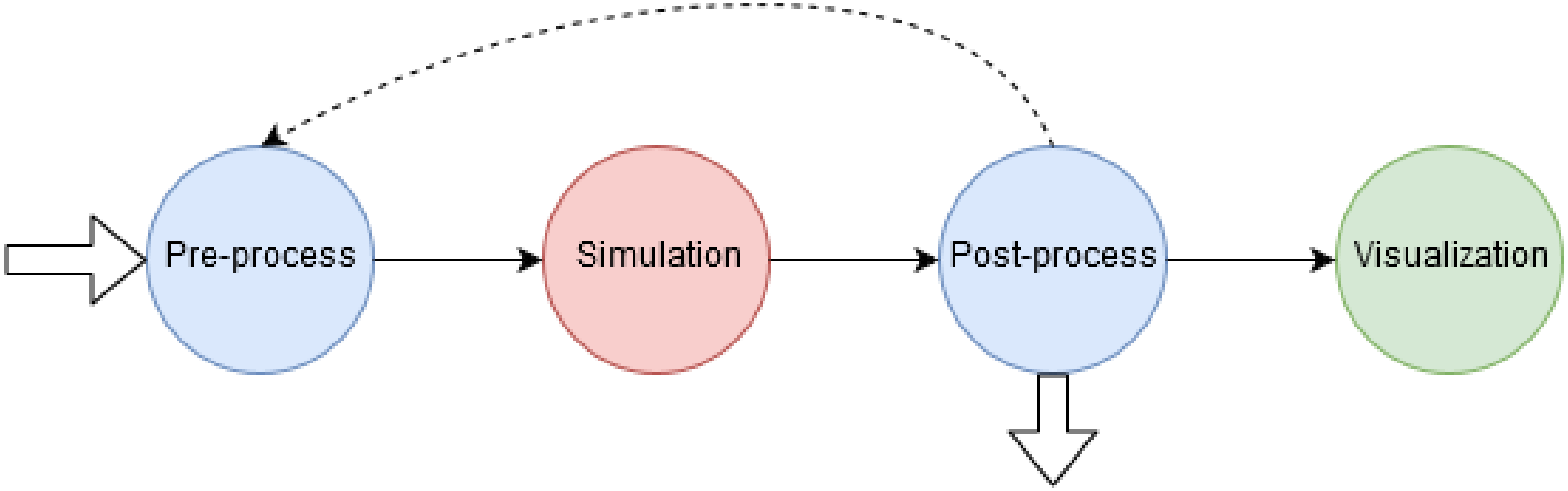


- + Performance
- + Scalability

- - Access
- - Provision
- - Ease of Usage

Why HPC & Cloud?

Maths workflow model



Why HPC & Cloud?

Steps Analysis

	Pre/Post	Simulation	Visualization
Features	❖ Small number of cores ❖ Small communication between processes ❖ Not HPC efficient	❖ Many cores, don’t fit in one node ❖ Heavy communication between processes	❖ Long-time running tasks ❖ Small number of cores ❖ Small communication between processes
Examples	• Data Movements • Big Data • Meshing	• Feel++ • FEniCS • OPM • Gromacs	• Paraview • SALOME

Proposed Solution

Key-value proposition:

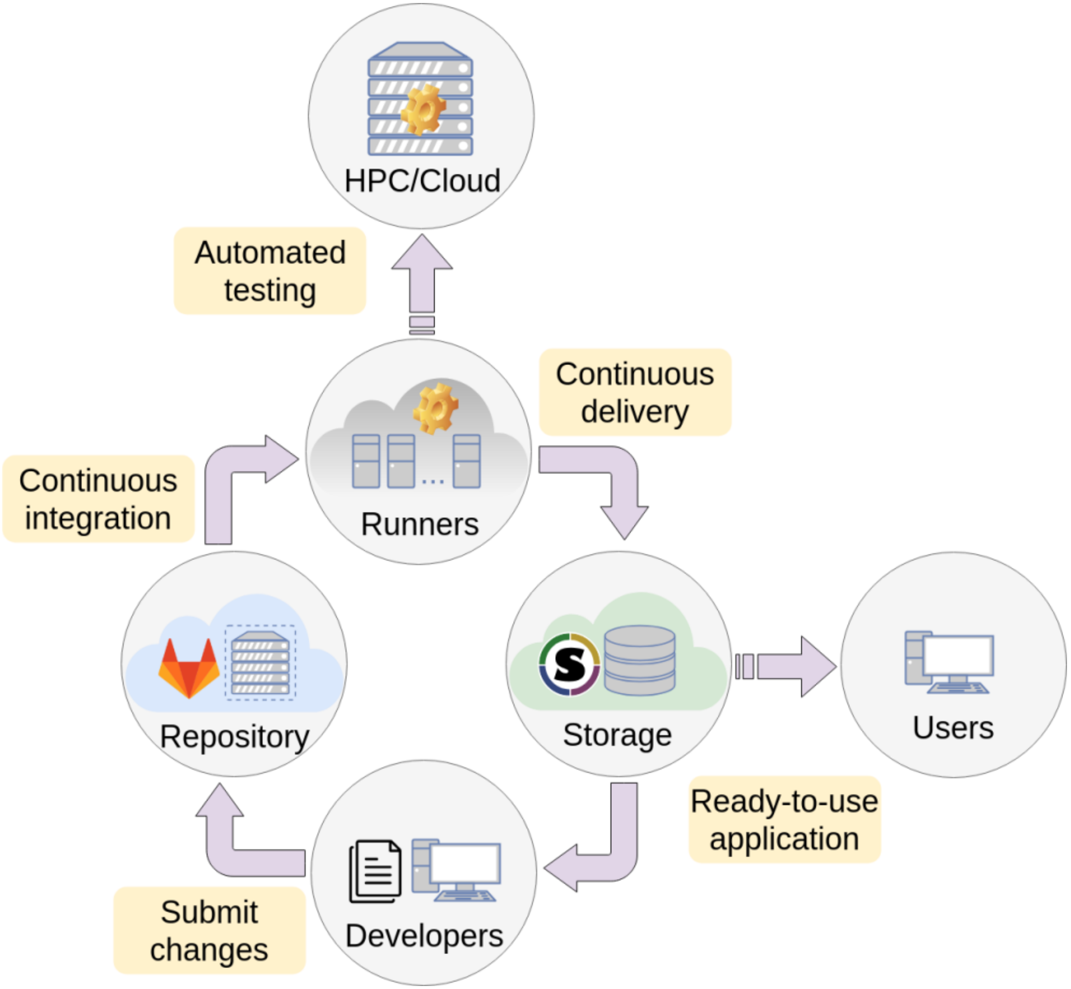
- Get full potential of both HPC and Cloud
- Automation: Encapsulation, CI/CD, Orchestration, Federation, Software as a service
- No vendor specific
- Open Source: Easy adoption & extensible

Implementation needs:

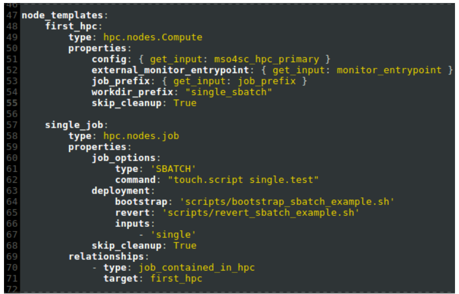
- DSL following TOSCA
- Two orchestration layers: “Meta-scheduler”
- Agentless architecture for HPC infrastructures monitoring
- Singularity containers

Proposed Solution

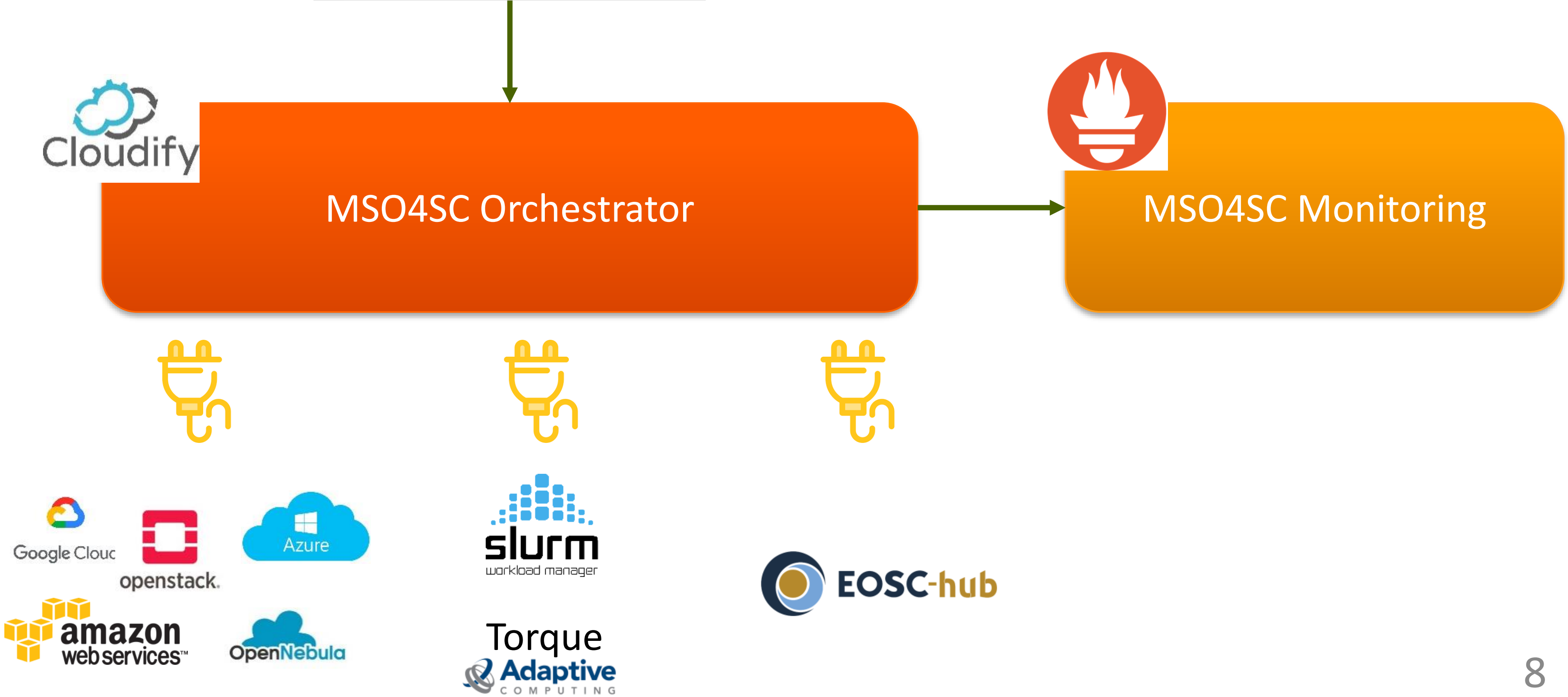
- CI/CD/CD
  - Code
  - Containerize
  - Test
  - Deliver
  - Ready to use!



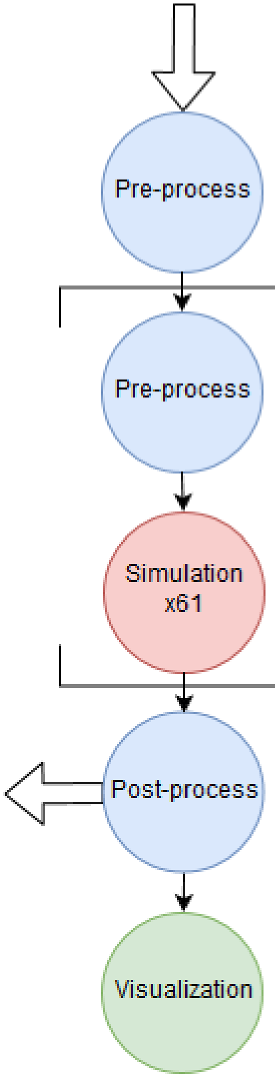
Proposed Solution

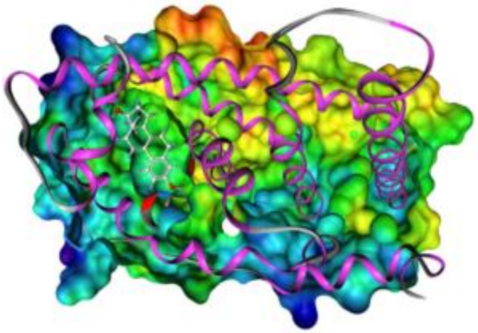


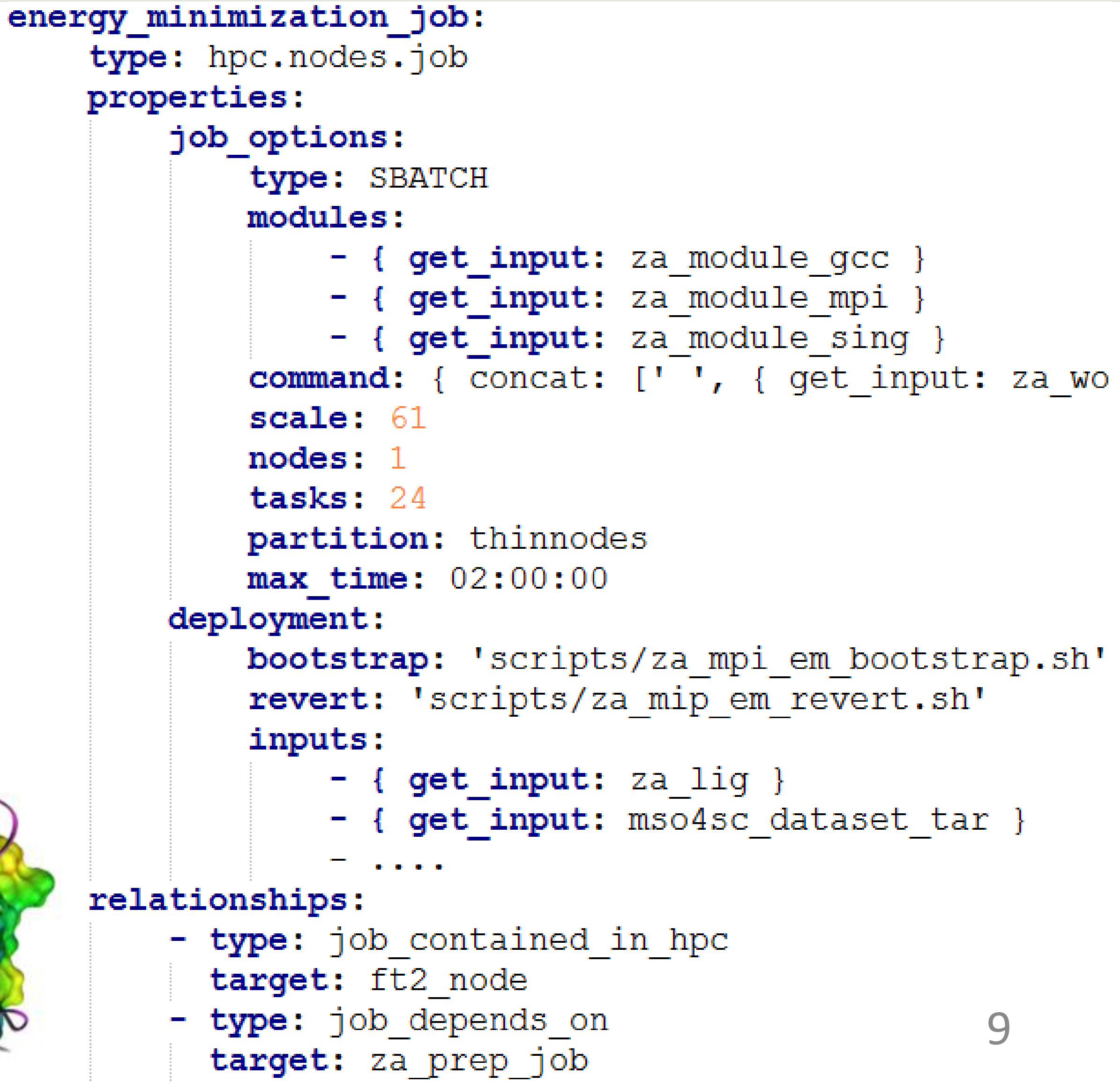
```
node_templates:
    first_hpc:
        type: hpc.nodes.Compute
        properties:
            config: { get_input: mso4sc_hpc_primary }
            external_monitor_entrypoint: { get_input: monitor_entrypoint }
            job_prefix: { get_input: job_prefix }
            workdir_prefix: "single_sbatch"
            skip_cleanup: True

    single_job:
        type: hpc.nodes.job
        properties:
            job_options:
                type: 'SBATCH'
                command: "touch.script single.test"
            deployment:
                bootstrap: 'scripts/bootstrap_sbatch_example.sh'
                revert: 'scripts/revert_sbatch_example.sh'
                inputs:
                    - 'single'
            skip_cleanup: True
        relationships:
            - type: job_contained_in_hpc
              target: first_hpc
```



Use case: ZibAffinity







```
energy_minimization_job:
    type: hpc.nodes.job
    properties:
        job_options:
            type: SBATCH
            modules:
                - { get_input: za_module_gcc }
                - { get_input: za_module_mpi }
                - { get_input: za_module_sing }
            command: { concat: [' ', { get_input: za_wo
            scale: 61
            nodes: 1
            tasks: 24
            partition: thinnodes
            max_time: 02:00:00
        deployment:
            bootstrap: 'scripts/za_mpi_em_bootstrap.sh'
            revert: 'scripts/za_mip_em_revert.sh'
            inputs:
                - { get_input: za_lig }
                - { get_input: mso4sc_dataset_tar }
                - ....
    relationships:
        - type: job_contained_in_hpc
          target: ft2_node
        - type: job_depends_on
          target: za_prep_job
```

Use case: ZibAffinity

Single executable Vs TOSCA DSL Blueprint

- Single executable
  - 24x61 cores during 40 minutes, wasting 23x61 cores during 5:44 minutes (mono core tasks)
  - Over 80 hours to compile and deploy on a new HPC
- TOSCA blueprint:
  - 17.03% core/h improvement (didn’t waste any resources)
  - Around 30 hours to build the blueprint and the containers. Runnable in multiple infrastructures.
  - Inputs: much easier to execute by a non expert user.

The TOSCA blueprints usage improves the resources efficiency, but the allocation time in the HPC *may* increase.

Use case: ZibAffinity

	Pre/Post	Simulation	Total
Exec Time	HPC ≈ Cloud	HPC < Cloud	46’ / 99’ | 46.82% HPC < Cloud HPC
Alloc Time	HPC ≈ Cloud	HPC > Cloud	646’’ / 77’’ | 88% HPC > Cloud Cloud
Total Time	HPC > Cloud 80%	HPC > Cloud 25.7%	186’ / 115’ | 38% HPC > Cloud Cloud
Core/h	HPC ≈ Cloud	HPC < Cloud 55%	HPC < Cloud| 34,47%

Atos Croupier:

- 143’ total time, 23.34% improvement over pure HPC execution.
- 48.3 % core/h improvement over pure cloud.

The orchestrator reverts *and improves* the overall time lost by using blueprints, while keeping the core/h optimizations.

Conclusions and Future Work

- Mixing both HPC & Cloud may improve the execution (depending on data size):
  - In core/hours spent
  - In total time to run
  - Releasing valuable HPC resources
- Different scheduling algorithms can be applied to optimize different aspects of a simulation
- *Smarter resource allocation, Improve data management, reconfiguration on the fly*
- *Test and compare with HPC in Cloud (i.e. through GoogleCloud)*
- *Create the VO for mathematics at EOSC*

MSO4SC

- Web: www.mso4sc.eu
- Docs: http://book.mso4sc.cemosis.fr
- Source: https://github.com/MSO4SC

- Follow us @mso4sc

- Contact us if you want to try!

Thank you for your attention!

Contact information:

Scientific Coordinator:	Zoltán Horváth (SZE) horvathz@math.sze.hu
Project Coordinator:	Francisco Javier Nieto (ATOS) francisco.nieto@atos.net
Website:	www.mso4sc.eu

MSO4SC

Mathematical Modelling, Simulation and Optimization for Societal Challenges with Scientific Computing

European Commission

Horizon 2020
European Union funding
for Research & Innovation

Grant agreement No. 731063

Atos

Research Center MATHEON
Mathematics for Key Technologies

UNIVERSITÉ DE STRASBOURG

cemosis
Centre de modélisation et de simulation de Strasbourg

SINTEF

bcam
basque center for applied mathematics

SZE - GYŐR

ZIB

KTH
VETENSKAP OCH KONST

CESGA

EU MATHS IN