# EASILY DEPLOYING OPENFOAM APPLICATIONS WITH PYTHON AND CLOUD HPC

RODRIGO VALÉRIO[1], HUGO PENEDONES[1], LUÍS SARMENTO[1]
[1]*Inductiva Research Labs {rvalerio, hpenedones, sarmento}@inductiva.ai*

***Keywords:*** *OpenFOAM, Python, Cloud-HPC, Inductiva API*

We present a procedure for OpenFOAM experts to share their modeling work with a broader audience, including non-experts, potentially creating new monetization opportunities. For example, machine learning researchers and engineers are increasingly interested in generating large datasets of physical simulations to train neural networks.

This approach uses the Inductiva API [1] to manage OpenFOAM simulations in the cloud transparently and scalably. The pre-processing and post-processing logic is encapsulated in a simple Python package that can be easily distributed and installed by users. By abstracting the complexities of CFD modeling and the underlying computational infrastructure, users can programmatically specify combinations of inputs and parameters to explore and easily obtain results from thousands of independent simulations running parallel.

**Packaging your OpenFOAM application**

As an example, we take the OpenFOAM *Motorbike* tutorial for incompressible fluids [2], and encapsulate it inside a standalone Python package using the Inductiva API. The final interface allows users to run multiple parallel OpenFOAM simulations without needing in-depth CFD knowledge, access to HPC clusters, or even a local OpenFOAM installation.

1. **Define OpenFOAM Input Files**: Put all the required input files for OpenFOAM in a directory called "templates". The *motorbike* tutorial contains the `0`, `constant`, and `system` sub-directories with the appropriate initial and boundary conditions, physical properties, and solver settings.
2. **Make relevant parameters configurable**: Change the OpenFOAM input files so that some simulation parameters of interest become variables, to be filled-in by the *inductiva.TemplateManager*. In this case, the wind speed variable will be an input of the Python program, while the wheelbase length and the projection area of the object will be automatically computed based on the 3D mesh of the object that is placed inside the wind tunnel.

```
dragDir          (1 0 0);
CofR             (1 0 0);  // Axle midpoint on ground
pitchAxis        (0 1 0);
magUInf          {{ wind_speed }};    // Wind-speed
lRef             {{ length }};        // Wheelbase length
Aref             {{ area }};          // Object projection area
```

**Figure 1: Section of the system/forceCoeffs file.**

3. **Define Python interface**: Implement a simple "WindTunnel" class, whose dimensions will be specified in the constructor, and define a method to insert a 3D object mesh inside the wind tunnel. Finally, define a method to start the simulation based on additional parameters such as the wind speed, or the number of iterations.
4. **Simulation Management with Inductiva API**: Employ the Inductiva API to run and manage OpenFOAM simulations. These will be executed on demand in dedicated cloud Virtual Machines (VMs).

See the full open-source code at: `https://github.com/inductiva/wind-tunnel`.

### Running multiple parallel simulations

In a few lines of Python code, the user can now invoke the WindTunnel package and run multiple simulations in parallel for different 3D meshes and wind speeds. This allows non-CFD experts, such as those in the machine learning community, to generate large datasets of simulations to train neural networks for predicting physical quantities like the drag coefficient.

```
import windtunnel

wind_tunnel = windtunnel.WindTunnel(dimensions = (20, 10, 8))
for fname in ["car", "excavator", "race_car", "truck"]:
    wind_tunnel.set_object(object_path=f"assets/{fname}.obj")

    for wind_speed_ms in [10, 30, 50]:
        task  = wind_tunnel.simulate(wind_speed_ms,
                                     num_iterations=300,
                                     resolution=5)
```

**Figure 2: Python code for running multiple parallel simulations for different objects and wind speeds.**

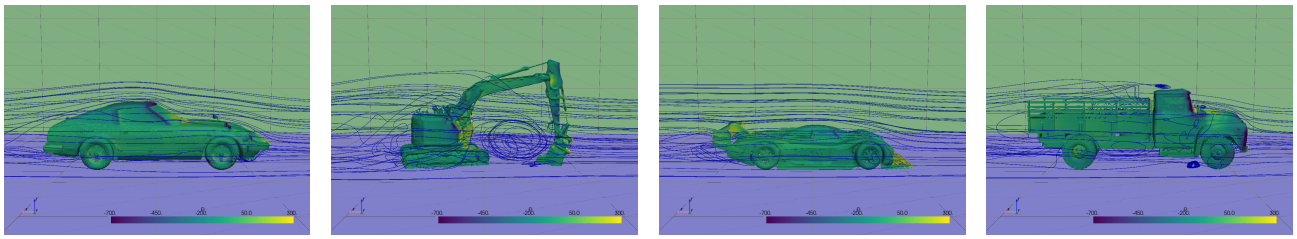

**Figure 3: Streamlines and pressure maps for different vehicles with a wind speed of 30m/s.**

### Dedicated Hardware

Users can launch a pool of dedicated, non-communicating VMs that can run multiple simulations in parallel:

```
machine_group = inductiva.resources.MachineGroup(
                    machine_type="c2d-highcpu-32",
                    num_machines=4,
                    spot=True)
machine_group.start()
```

**Figure 4: Python code to launch dedicated VMs with the Inductiva API.**

The machine_type follows Google Cloud's naming convention, e.g., *c2-standard-16*, specifies the CPU architecture, number of virtual CPUs and RAM memory. Spot instances are cheaper VMs that can be preempted by the cloud provider.

### Acknowledgments

We thank Ivan Pombo and Luís Cunha for their contributions to the Inductiva Wind Tunnel Python package.

### References

[1] Inductiva python api for large-scale simulation. [Online]. Available: https://github.com/inductiva/inductiva

[2] Openfoam motorbike tutorial. [Online]. Available: https://github.com/OpenFOAM/OpenFOAM-8/tree/master/tutorials/incompressible/simpleFoam/motorBike