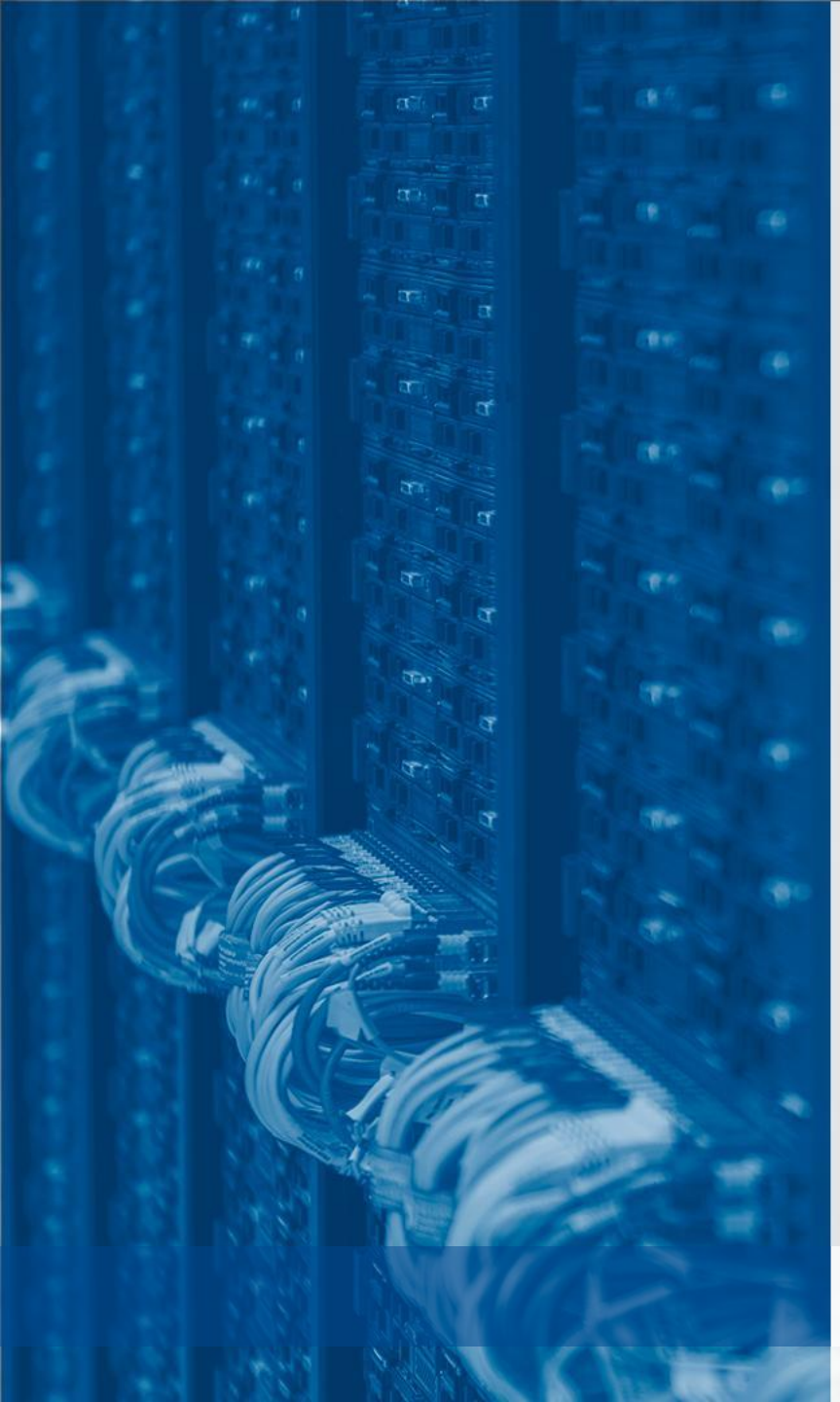


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Profiler integration in Autosubmit 4

Pablo Goitia González

Computational Earth Sciences Group | August 4th, 2023 Meeting

# Why Autosubmit needs a profiler?

- Measure the performance of Autosubmit.
- Detect bottlenecks that hinder performance.
- Reduce energy consumption.
- Improve code quality, in general.

## Find the docs here! 👇

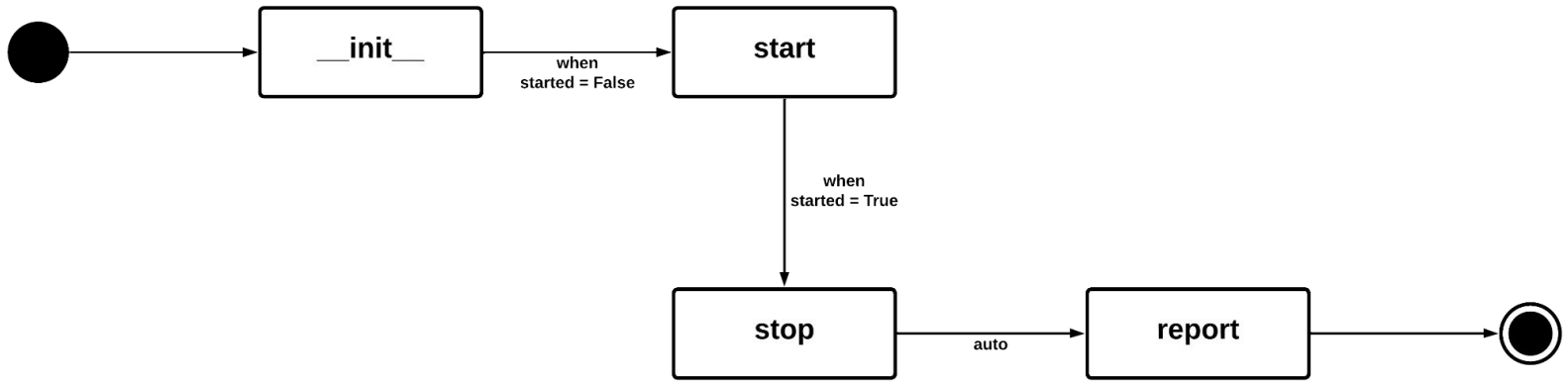
https://autosubmit.readthedocs.io/en/profiler/userguide/run/profiler.html

# Fundamentals

- Key: Make it as simple as possible.
- Mainly uses *cProfile*, *pstats* and *psutil* libraries.
  https://docs.python.org/3/library/profile.html
  https://psutil.readthedocs.io/en/latest/
- Assumable overhead.
- *cProfile* works on a single thread.

# How it works? Scope.

- The structure of the code works as a simple state machine.




Specially designed to be executed with the *autosubmit run* command!

# How can I profile myself?

**Step 1:** Make sure that your AS version integrates the profiler (AS>v4.0)

**Step 2:** Run the *autosubmit run* command as you always did, but adding the *--profile* or *-p* flag, as follows:

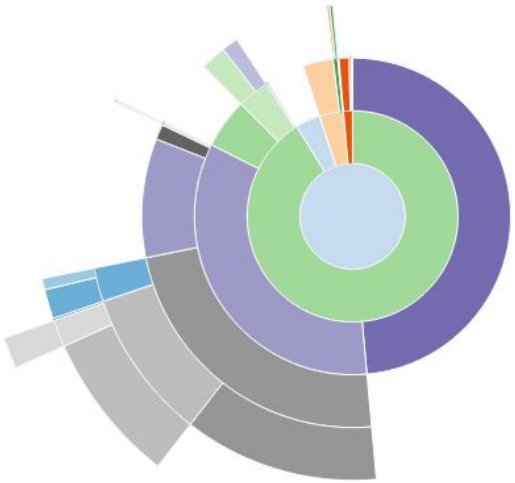
*autosubmit run --profile <EXPID>*

- You will receive a full performance report on the console.
- This report will also be stored in the <EXPID>/tmp/profile directory in two separate files, a text file and a .prof binary that can be manipulated with external tools.

# Post-processing

## Want to see or manipulate the contents of the .prof file?

- There are many tools to do it! Pstats, for example.
- But I highly recommend you to use SnakeViz.
  https://jiffyclub.github.io/snakeviz/



Search:

| ncalls | tottime | percall | cumtime | percall | filename:lineno(function) |
|---|---|---|---|---|---|
| 1 | 0.000421 | 0.000421 | 0.000421 | 0.000421 | ~:0(<built-in method listdir>) |
| 1 | 0.000104 | 0.000104 | 0.000202 | 0.000202 | functools.py:441(wrapper) |
| 1 | 7.9e-05 | 7.9e-05 | 0.000294 | 0.000294 | fnmatch.py:48(filter) |
| 1 | 6.7e-05 | 6.7e-05 | 8e-05 | 8e-05 | functools.py:342(_make_key) |
| 1 | 4.4e-05 | 4.4e-05 | 0.00079 | 0.00079 | glob.py:61(glob1) |

Images from https://jiffyclub.github.io/snakeviz/

# How to profile a specific section of code?

## AUTOSUBMIT RUN FUNCTION

**Reminder:** This profiler's scope is the *autosubmit run* command.

### Remember the status diagram. Just place in the code…

```
if profile:
    profiler = Profiler(expid)
    profiler.start()
```

[more tons of code…]

```
if profile:
    profiler.stop()
```

Instantiate the profiler, passing the EXPID as parameter, and start the execution.

Terminate the execution, preferably in a section of code that always executes even if a critical failure occurs, such as within a finally clause.

# Yes, but… What if I want to profile a section out from the “run” function?

**BE CAREFUL!**

You are a programmer, there is (almost) nothing impossible for you, but keep in mind that this profiler was made specially for the *Autosubmit Run* function.

## That means…

The profiler inherits the cProfile library characteristics, and that means that works on a single thread. If you use it over multiple threads or synchronized sections, the results will be far from accurate.

The EXPID restriction: the profiler requires your the ID of your experiment to know where to save the output files. You could bypass this restriction just passing as parameter another string when you plan to run it in a function that does not use it.


Demo time! :)


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Thanks!

pablo.goitia@bsc.es | Models and Workflows Team