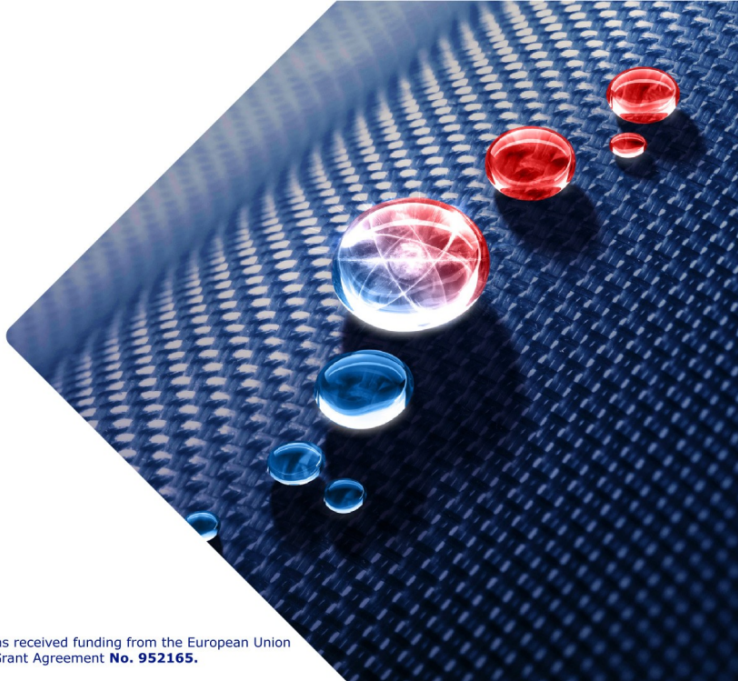


Introduction to Verificarlo CI

Aurélien Delval

November 21, 2022



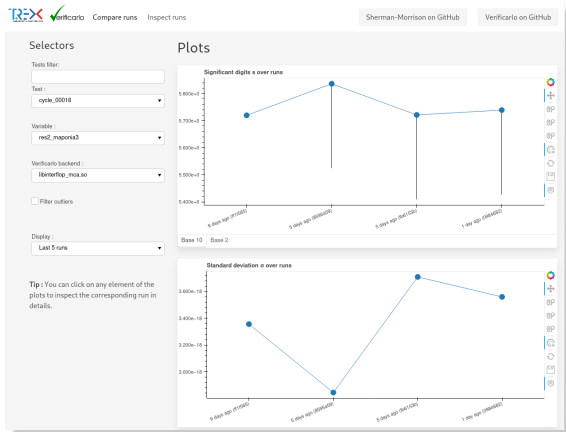
Verificarlo CI is a tool built on top of Verificarlo that lets its user :



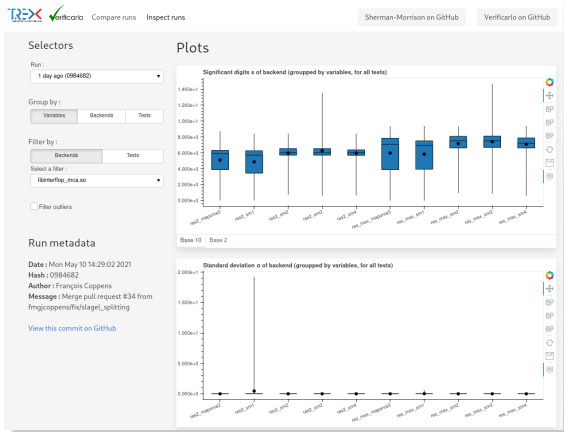
- run
- automatize
- visualize

... custom Verificarlo tests.

These tests can be automatized with **Github Actions** or **Gitlab CI/CD**, and their results can then be visualized and shared in **HTML reports**.



- Visualize evolution of **one variable over time** (commits)
- Shows significant digits s , standard deviation σ , variable distribution

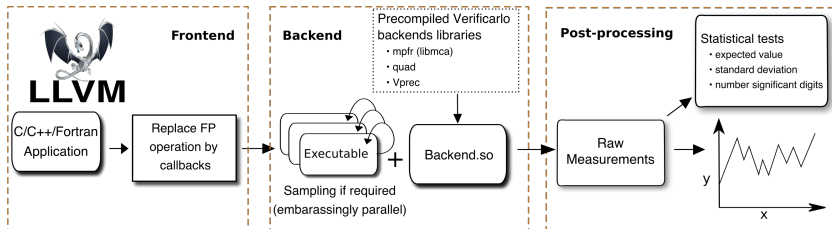


- Focus in depth on **one particular run**
- Different group by / filter by options
- For each group, shows distribution of s , σ and compute aggregated variable average

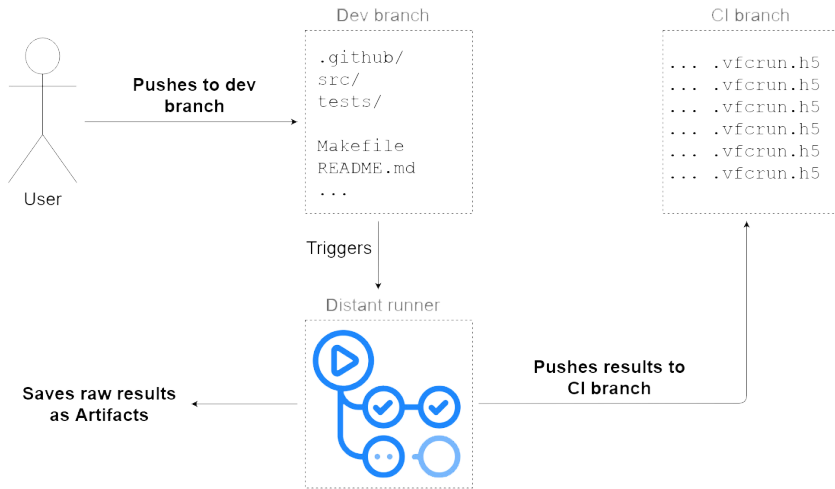
The CI integration pipeline relies on two different git branches :

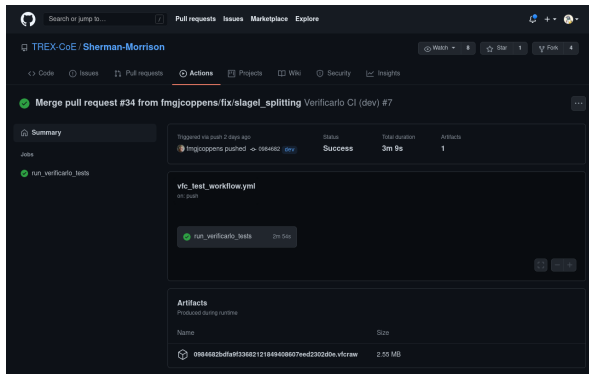
- the **"dev" branch**, which is pre-existent and contains the project's source code
- the **"CI" branch**, which will be automatically be updated with result files, and is completely orphan of the dev branch

Note that while the tool has been **designed to integrate with git and CI systems**, it can still be used **completely independently**, simply as a way to visualize test results.



Verificarlo CI is integrated into Verificarlo as a **post-processing** tool.





- Workflow runs will be triggered **after each push** by default
- The CI branch will be **updated automatically**
- Raw results will be accessible as **Github Artifacts**
- The CI setup can be automated with the ***vfc_ci setup*** command

In order to export test data from the programs to the tool itself, Verificarlo CI uses a **probes** system which allows the user to :

- **define test variables** (or *probes*) identified by a unique test name / variable name combination
- associate them with an **optional accuracy target** (absolute or relative)

... and **interfaces with Verificarlo CI** itself.

- Here is a basic example where we want to store the value of `var` in a probe called `varName` belonging to a test `testName` in C:

```
vfc_probe(&probes, "testName", "varName", var);
```

- A Fortran interface is also provided using the *ISO_C_BINDING* module:

```
vfc_probe(probes,  
"testName//C_NULL_CHAR",  
"varName//C_NULL_CHAR",  
var);
```

Verificarlo CI is accessed through a single command-line interface, *vfc_ci*. It provides 3 different subcommands :

- *setup* : an helper script to initialize the **CI workflow**
- *test* : start a **test run**
- *serve* : launch a server giving access to the **HTML report**

The `vfc_ci test` command manages the tests runs. It is configured with the `vfc_tests_config.json` file :

```
{
  "make_command": "make tests",
  "executables": [
    {
      "executable": "bin/test",
      "parameters" : "foo",
      "vfc_backends": [
        {
          "name": "libinterflop_mca.so --precision=53",
          "repetitions": 20
        }
      ], [...]
    }
  ]
}
```

After executing the tests and gathering the results, the **postprocessing pipeline** is executed. For stochastic backends, we compute for each probe :

- the **empirical average** μ and the *standard deviation* σ
- the **number of significant digits** s_2, s_{10}
- the distribution's **quantiles**

The accuracy checks (with a target t) are evaluated as follows :

- if the check is **absolute**, $\sigma < t$?
- if the check is **relative**, $\frac{\sigma}{|\mu|} < t$?

- The **Verificarlo CI tutorial**, can be found at [*https://github.com/verificarlo/vfc_ci_tutorial*](https://github.com/verificarlo/vfc_ci_tutorial)
- For more details, the **Verificarlo CI documentation** can be found at [*https://github.com/verificarlo/verificarlo/blob/master/doc/06-Postprocessing.md#verificarlo-ci*](https://github.com/verificarlo/verificarlo/blob/master/doc/06-Postprocessing.md#verificarlo-ci)
- If you plan to use either Singularity or Docker, you will need to run the following script just before starting your server to update some Python packages (follow the instructions): [*https://gist.github.com/PurplePachyderm/2ae9fedf7f458bdae7534adc0d5862e0*](https://gist.github.com/PurplePachyderm/2ae9fedf7f458bdae7534adc0d5862e0)

- If you plan to use the **Singularity image on CALMIP** while connecting with the VPN, you can set up port forwarding with :
> `ssh -L [port]:127.0.0.1:[port]`
> `-p 11300 [username]@127.0.0.1`
- Then, to load the Verificarlo Singularity image:
> `module load singularity`
> `singularity run /usr/local/trex/verificarlo/ver [...] .sif`

- If you plan to use the **Docker image**, you can download it and setup a container with port forwarding :

```
> docker pull verificarlo/verificarlo  
> docker run -it -p [port]:[port]  
verificarlo/verificarlo /bin/bash
```
- You can optionally set up a shared directory when creating the container:

```
-v [/path/to/host/dir]:[/path/to/docker/dir]
```