CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

ReFrame: A Regression Testing Framework Enabling Continuous Integration of Large HPC Systems

HPC Advisory Council 2018

Victor Holanda, Vasileios Karakasis, CSCS

Apr. 11, 2018

ReFrame in a nutshell

Regression Testing of HPC Systems

Why is it so important?

- Ensures quality of service
- Reduces downtime
- Early detection of problems

Regression Testing of HPC Systems

But it's a painful story

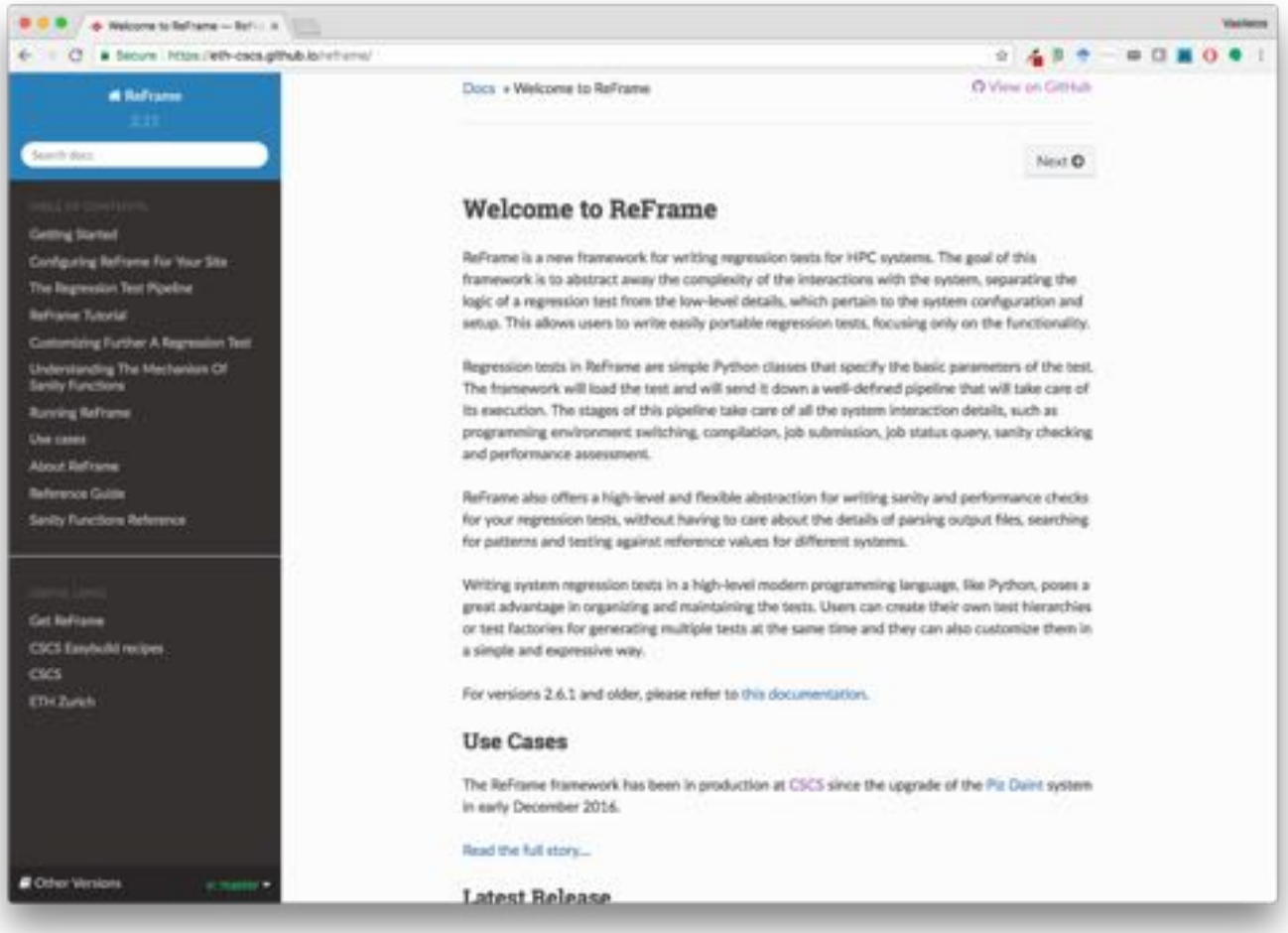
- In-house custom solutions per center
- Non portable monolithic regression tests
  - Tightly coupled to the system configuration and programming environments
- Large maintenance overhead
  - Replicated code of the system interaction details
  - Test's logic is lost in unrelated lower level details

Reluctance to implement new regression tests!

What Is ReFrame?

A new regression framework that

- allows writing **portable** HPC regression tests in Python,
- **abstracts away** the system interaction details,
- lets users **focus** solely on the logic of their test.



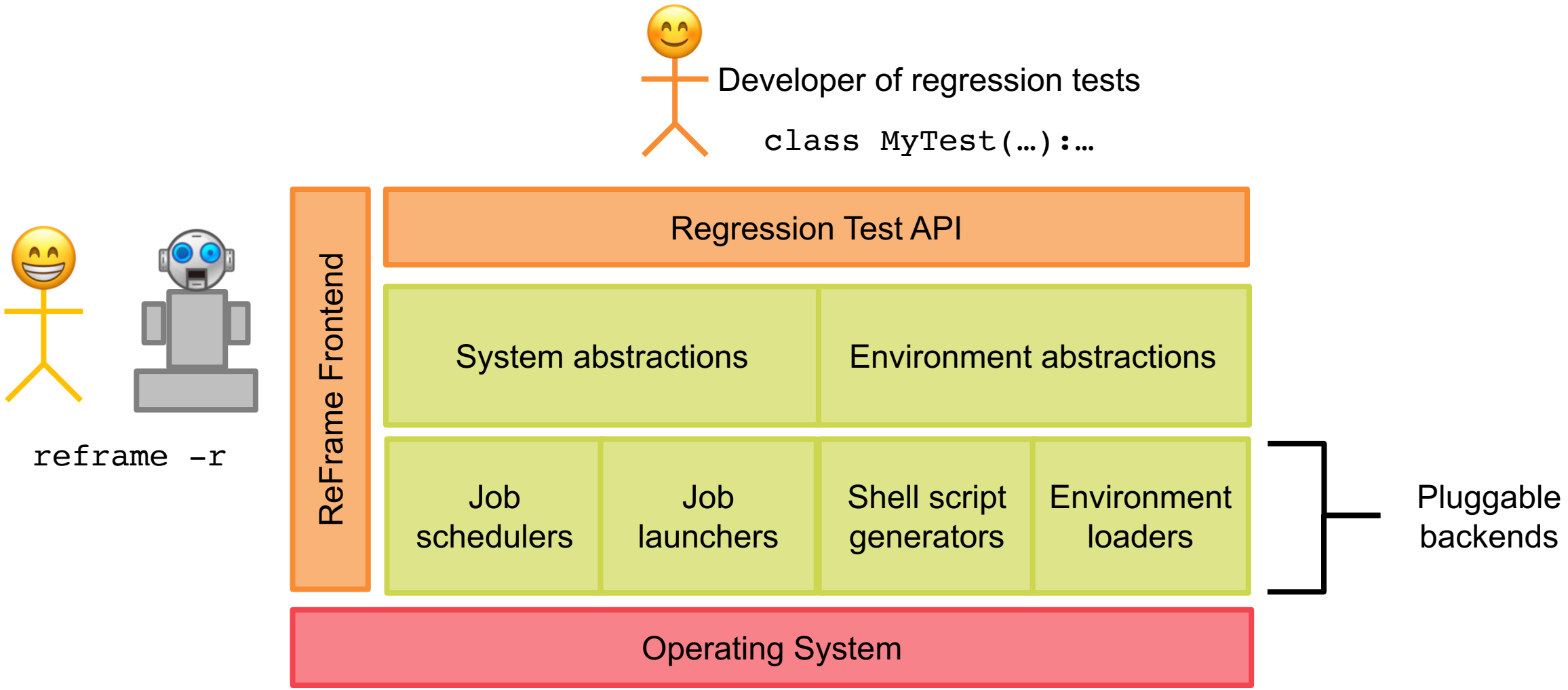
https://github.com/eth-cscs/reframe

Design Goals

- Productivity
- Portability
- Speed and Ease of Use
- Robustness

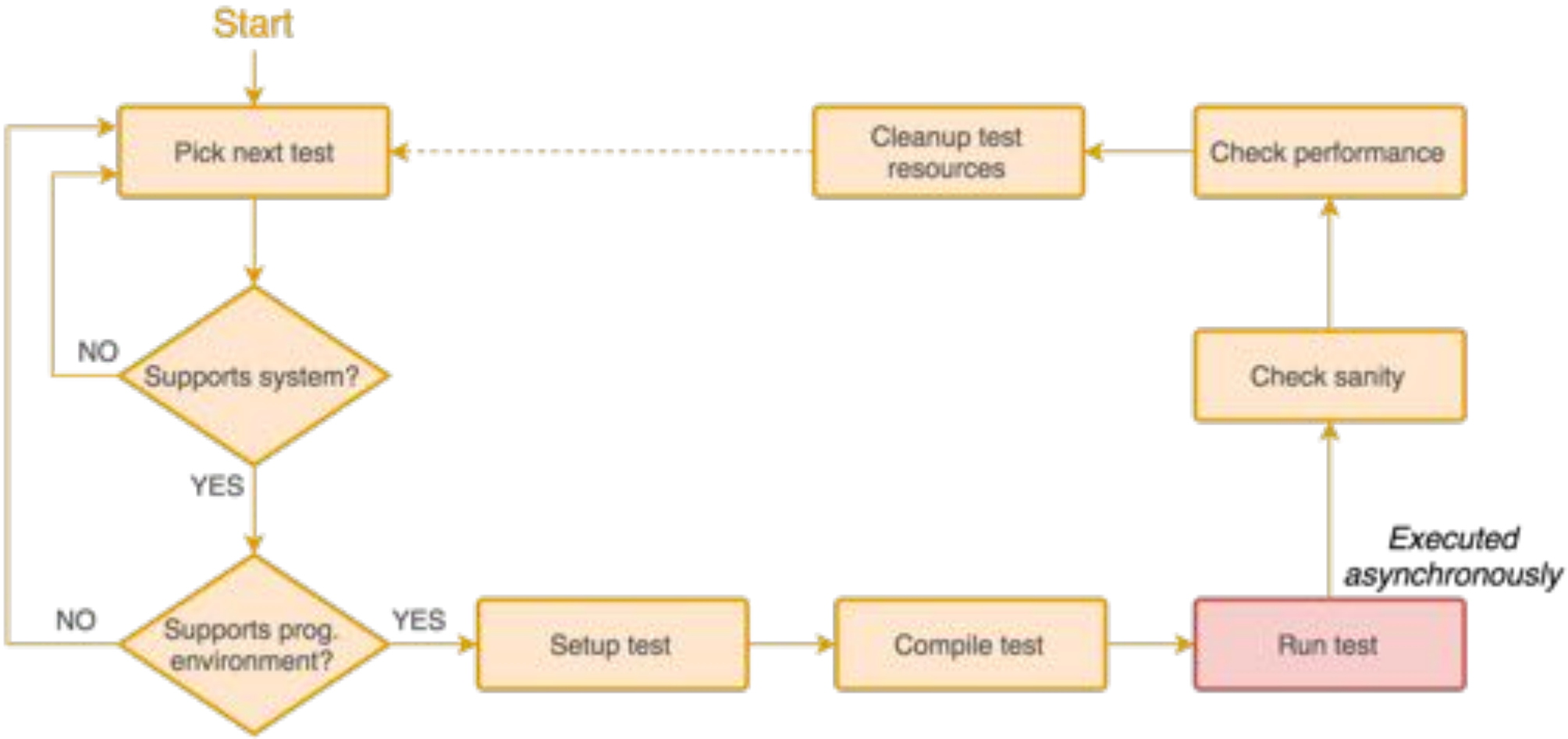
Write once, test everywhere!

ReFrame's architecture



The Regression Test Pipeline

A series of well defined phases that each regression test goes through



Some Features

- Support for Slurm (with and without srun) and simple batch scripts
- Support for different modules systems (Tmod, Lmod, no modules.)
- Seamless support of multiple prog. environments and HPC systems
- Flexible organization of the regression tests
- Progress and result reports
- Asynchronous execution of regression tests
- Complete documentation (tutorials, reference guide)
- And many more (*https://github.com/eth-cscs/reframe*)

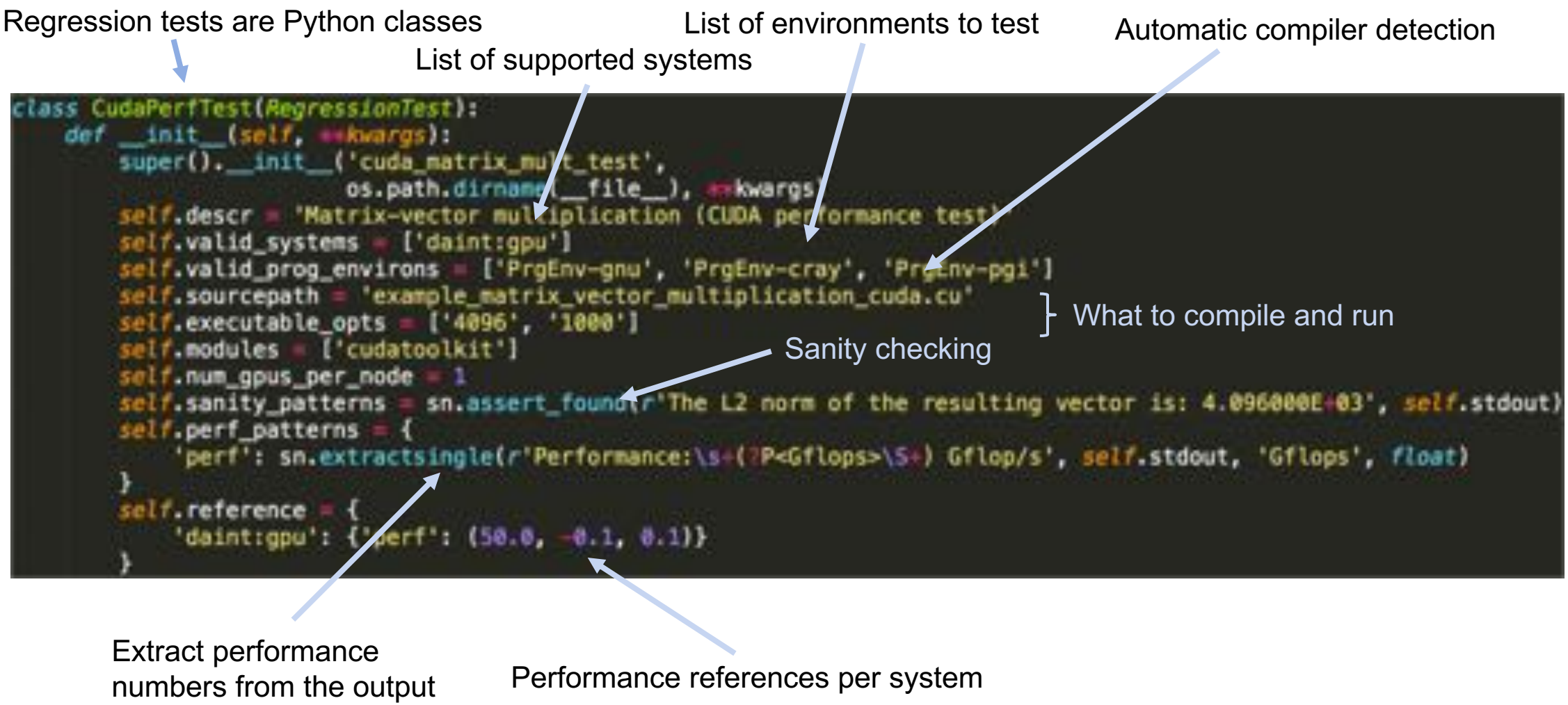
Writing a regression test in ReFrame

A regression test writer should not care about...

- How access to system partitions is gained and if there are any.
- How (programming) environments are switched.
- How its environment is set up.
- How a job script is generated and if it's needed at all.
- How a sanity/performance pattern is looked up in the output.

ReFrame allows you to focus on the logic of your test.

Writing a regression test in ReFrame




```
class CudaPerfTest(RegressionTest):
    def __init__(self, **kwargs):
        super().__init__('cuda_matrix_mult_test',
                         os.path.dirname(__file__), **kwargs)
        self.descr = 'Matrix-vector multiplication (CUDA performance test)'
        self.valid_systems = ['daint:gpu']
        self.valid_prog_environs = ['PrgEnv-gnu', 'PrgEnv-cray', 'PrgEnv-pgi']
        self.sourcepath = 'example_matrix_vector_multiplication_cuda.cu'
        self.executable_opts = ['4096', '1000']
        self.modules = ['cudatoolkit']
        self.num_gpus_per_node = 1
        self.sanity_patterns = sn.assert_found(r'The L2 norm of the resulting vector is: 4.096000E+03', self.stdout)
        self.perf_patterns = {
            'perf': sn.extractsingle(r'Performance:\s+(?P<Gflops>\S+) Gflop/s', self.stdout, 'Gflops', float)
        }
        self.reference = {
            'daint:gpu': {'perf': (50.0, -0.1, 0.1)}
        }
```

Running ReFrame

- Run tests sequentially:
  - `./bin/reframe -c /path/to/checks -r`
- Run tests asynchronously:
  - `./bin/reframe -c /path/to/checks --exec-policy=async -r`
- Test selection (by name, tag, prog. environment)
- Failure reports
- Configurable logging
- Performance logging → allows keeping historical data

Running ReFrame (sample output)

```
[==========] Running 1 check(s)
[==========] Started on Thu Mar 22 17:35:21 2018

[----------] started processing example7_check (CUDA matrixmul test)
[ RUN      ] example7_check on daint:gpu using PrgEnv-cray
[       OK ] example7_check on daint:gpu using PrgEnv-cray
[ RUN      ] example7_check on daint:gpu using PrgEnv-gnu
[       OK ] example7_check on daint:gpu using PrgEnv-gnu
[ RUN      ] example7_check on daint:gpu using PrgEnv-pgi
[       OK ] example7_check on daint:gpu using PrgEnv-pgi
[----------] finished processing example7_check (CUDA matrixmul test)

[  PASSED  ] Ran 3 test case(s) from 1 check(s) (0 failure(s))
[==========] Finished on Thu Mar 22 17:35:44 2018
```

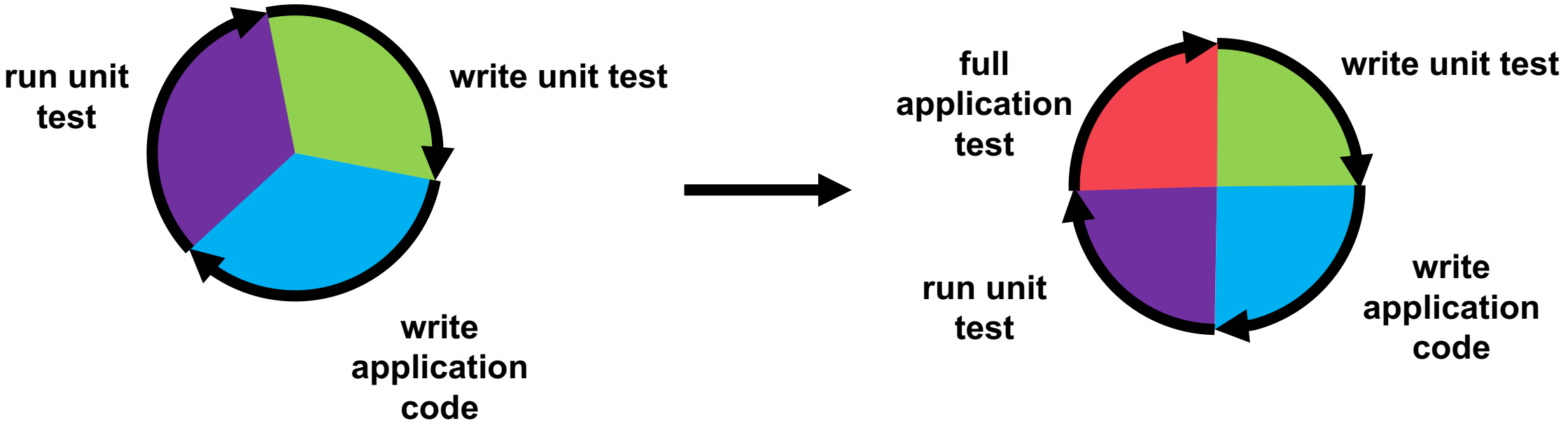
Running ReFrame (sample failure)

```
[==========] Running 1 check(s)
[==========] Started on Thu Mar 22 17:56:19 2018

[----------] started processing example7_check (CUDA matrixmul test)
[ RUN      ] example7_check on daint:gpu using PrgEnv-gnu
[     FAIL ] example7_check on daint:gpu using PrgEnv-gnu
[----------] finished processing example7_check (CUDA matrixmul test)

[  FAILED  ] Ran 1 test case(s) from 1 check(s) (1 failure(s))
[==========] Finished on Thu Mar 22 17:56:27 2018
==============================================================================
SUMMARY OF FAILURES
------------------------------------------------------------------------------
FAILURE INFO for example7_check
  * System partition: daint:gpu
  * Environment: PrgEnv-gnu
  * Stage directory: /path/to/stage/gpu/example7_check/PrgEnv-gnu
  * Job type: batch job (id=693731)
  * Maintainers: []
  * Failing phase: performance
  * Reason: sanity error: 49.244815 is beyond reference value 70.0 (l=63.0, u=77.0)
------------------------------------------------------------------------------
```

ReFrame inside a CI infrastructure

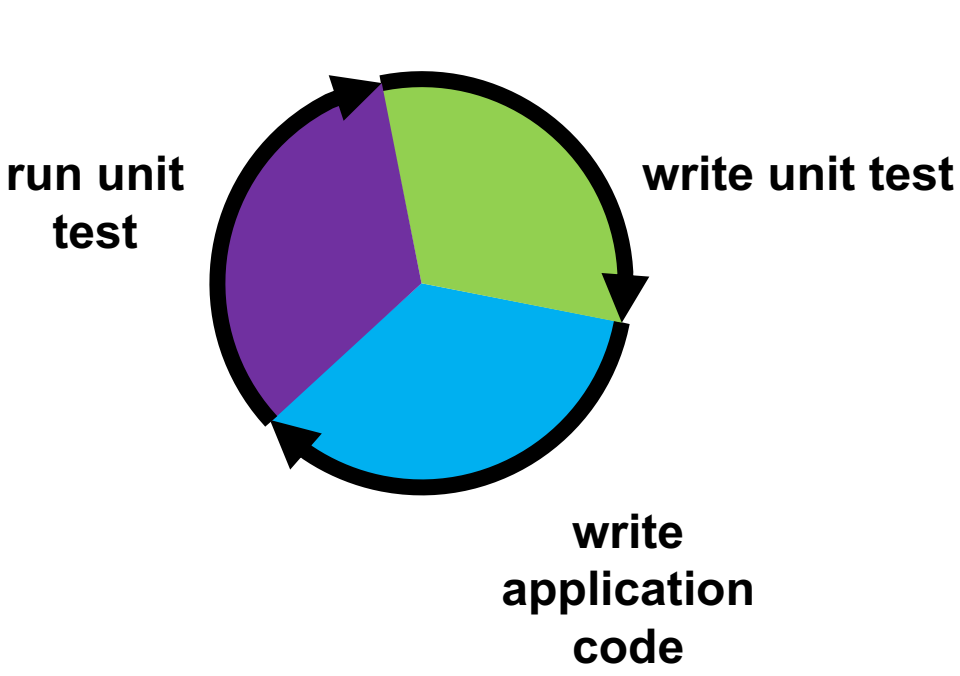
Running ReFrame as a CI tool for HPC applications

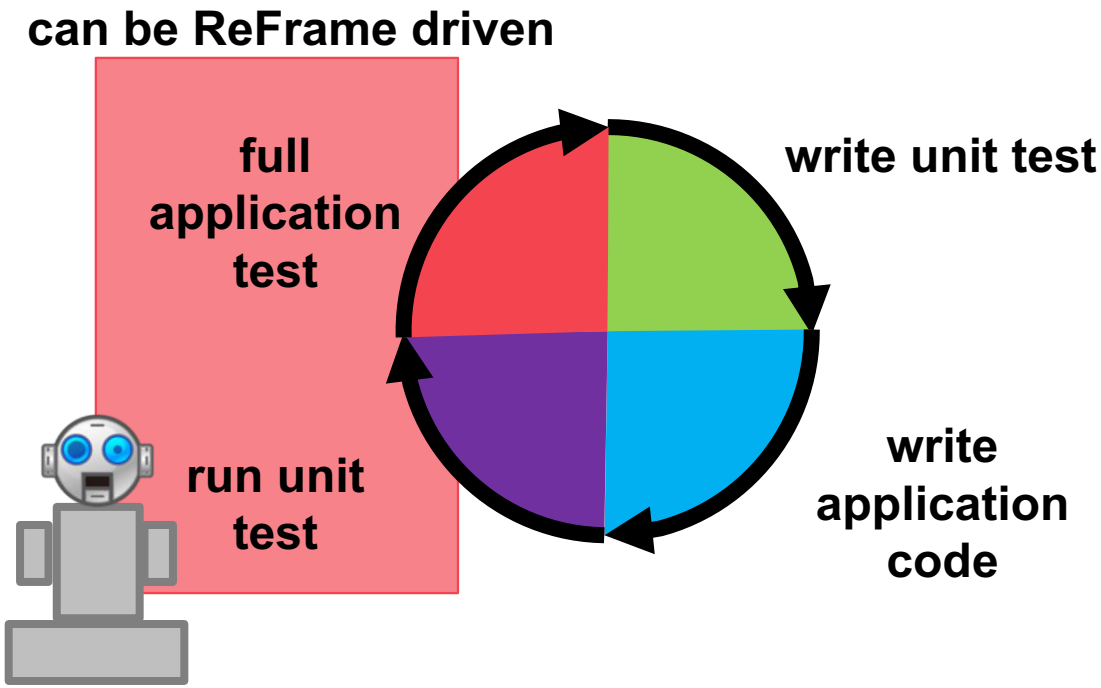
- Improve the development cycle of HPC applications
- Develop anywhere and test anywhere



Running ReFrame as a CI tool for HPC applications

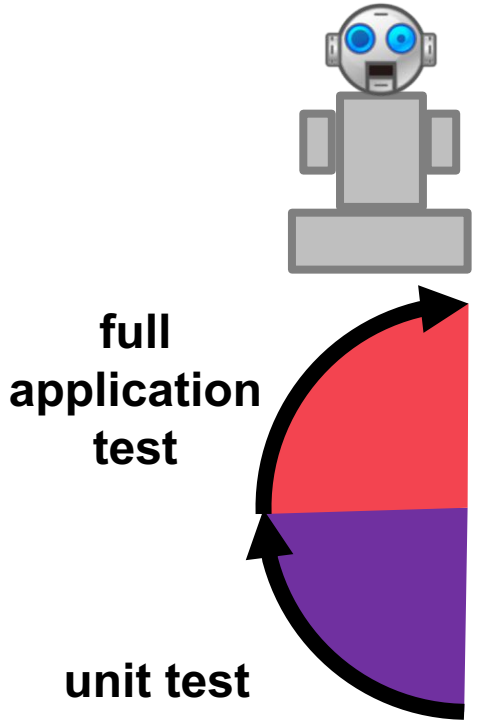
- Improve the development cycle of HPC applications
- Develop anywhere and test anywhere





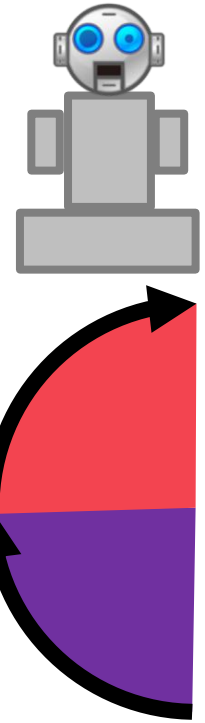
CI tool for HPC applications

- Login into different systems
- Loop over the proper programming environments
- Compile the code
- Create job scripts (if system has a queue system)
- Run unit tests
- Run different input files
- Collect sanity data
- Collect performance data
- Keep track if the code is still performing



CI tool for HPC applications

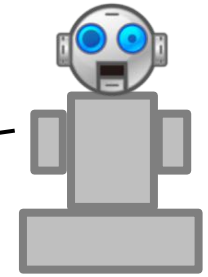
- Login into different systems
- Loop over the proper programming environments
- Compile the code
- Create job scripts (if system has a queue system)
- Run unit tests
- Run different input files
- Collect sanity data
- Collect performance data
- Keep track if the code is still performing



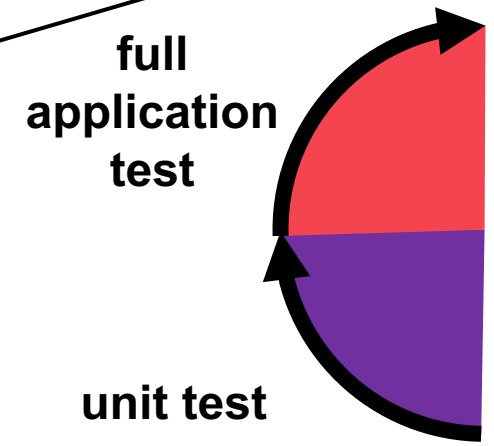
Support to send data to elastic search databases!

CI tool for HPC applications

- ~~Login into different systems~~
- Loop over the proper programming environments
- Compile the code
- Create job scripts (if system has a queue system)
- Run unit tests
- Run different input files
- Collect sanity data
- Collect performance data
- Keep track if the code is still performing



CI infrastructure



ReFrame

Support to send data to elastic search databases!

Running ReFrame as a CI tool for HPC applications

1. Add new **system** to ReFrame configuration inside your project.
2. Store your **ReFrame tests** in your project.
3. **Run** your tests on target system using ReFrame.

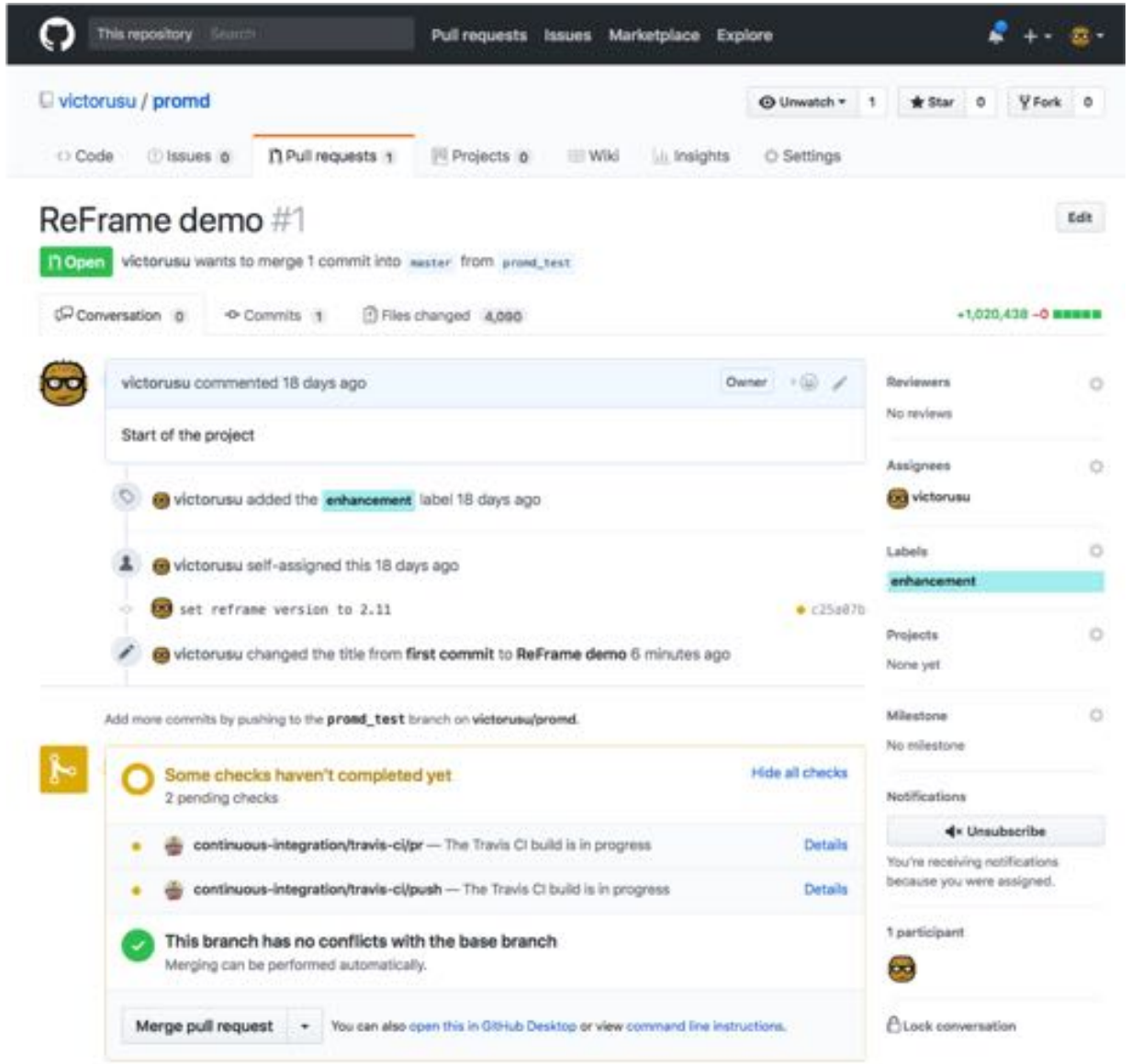
Use the same tests to run on Piz Daint, your laptop or a Travis VM!

Demo Time

1. Running ReFrame

2. Integration with TRAVIS (https://github.com/victorusu/promd/pull/1)

Travis – PROMD demo



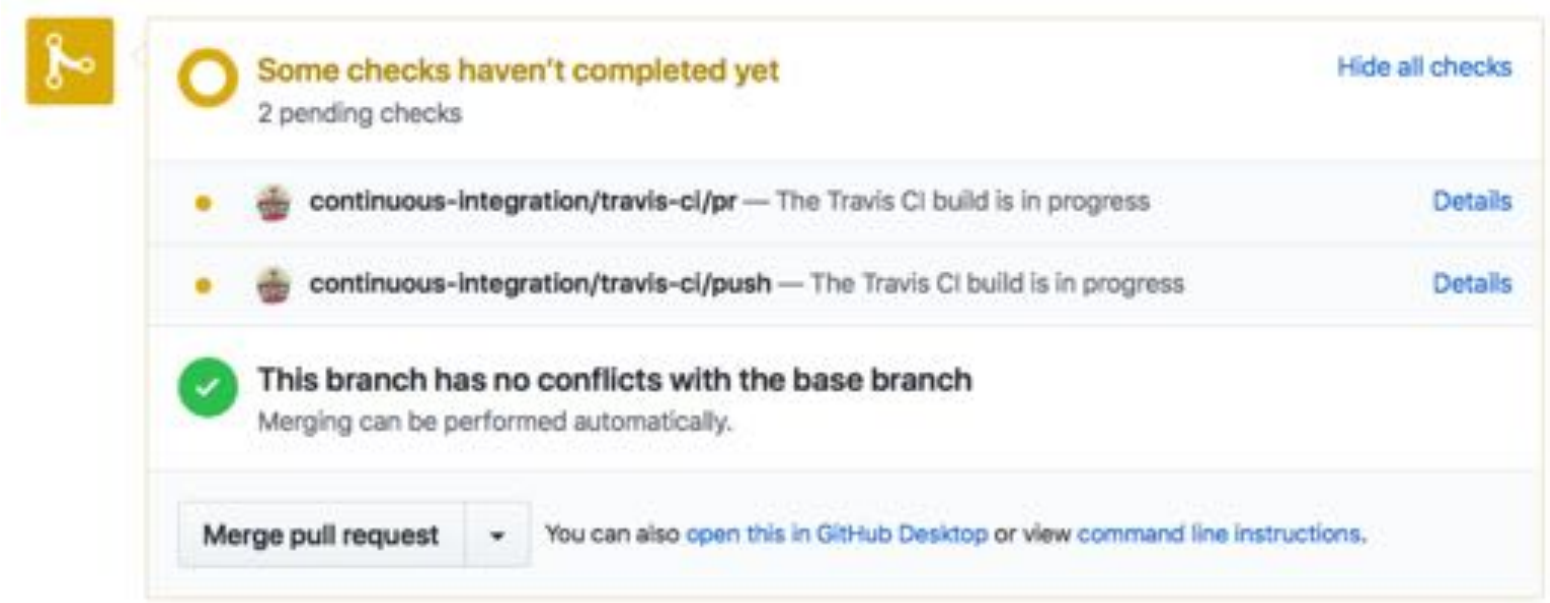
Travis – PROMD demo

217 ■■■■■ promd.py

View

```
@@ -0,0 +1,217 @@
+import itertools
+import os
+
+import reframe.utility.sanity as sn
+from reframe.core.pipeline import RegressionTest
+
+
+class PROMDCheck(RegressionTest):
+    def __init__(self, num_cores, variant, input, **kwargs):
+        super().__init__("promd_%s_%s_%s_check" % (input, num_cores, variant),
+                         os.path.dirname(__file__), **kwargs)
+
+        output_file = '%s.omd' % input
+        self.compiler_option = "omp" if  variant == 'omp' else "promd"
+        self.num_cores_avail = 4
+
+        #self.sourcepath = 'promd'
+        self.sourcesdir = 'promd'
+        self.keep_files = [output_file]
+
+        self.valid_systems = ['daint:gpu', 'generic']
+        self.descr = 'PROMD %s %s check with %s cpus ' % (input, variant, num_cores)
+
```

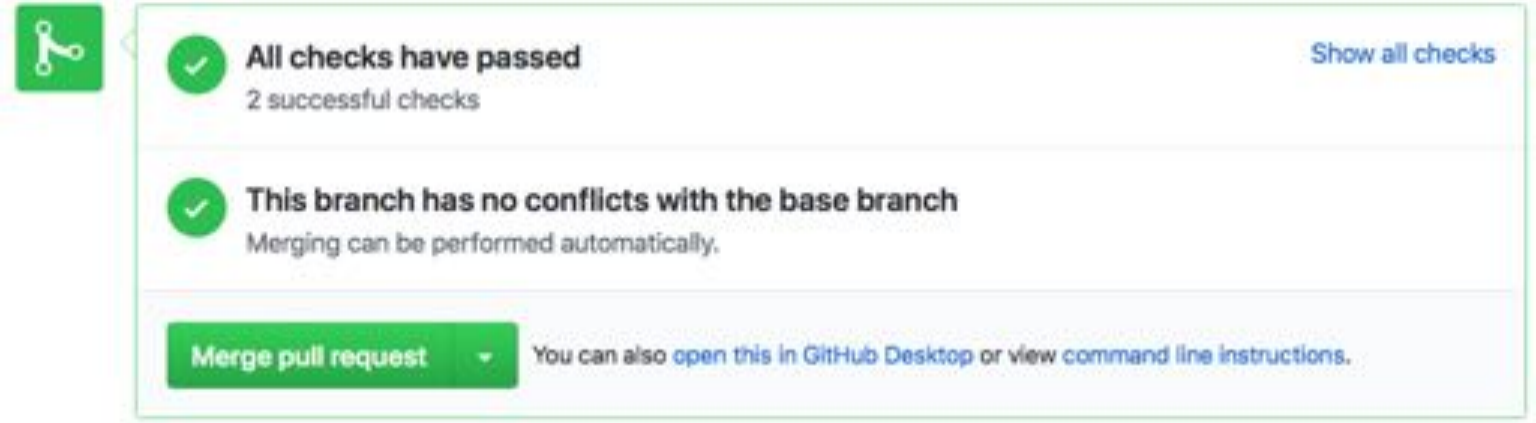
Travis – PROMD demo



Travis – PROMD demo

```
$ python3 ./reframe/reframe.py -c promd.py --system generic -r
Command line: ./reframe/reframe.py -c promd.py --system generic -r
Reframe version: 2.11
Launched by user: travis
Launched on host: travis-job-52ce71d7-d471-4cbd-8d28-67d708f2212a
Reframe paths
=============
    Check prefix      :
    Check search path : 'promd.py'
    Stage dir prefix  : /home/travis/build/victorusu/promd/stage/
    Output dir prefix : /home/travis/build/victorusu/promd/output/
    Logging dir       : /home/travis/build/victorusu/promd/logs
[==========] Running 9 check(s)
[==========] Started on Mon Apr  9 14:03:49 2018

[----------] started processing promd_methane_1_serial_check (PROMD methane serial check with 1 cpus )
[ RUN      ] promd_methane_1_serial_check on generic:login using builtin-gcc
[       OK ] promd_methane_1_serial_check on generic:login using builtin-gcc
[----------] finished processing promd_methane_1_serial_check (PROMD methane serial check with 1 cpus )

[----------] started processing promd_ethane_1_serial_check (PROMD ethane serial check with 1 cpus )
[ RUN      ] promd_ethane_1_serial_check on generic:login using builtin-gcc
```

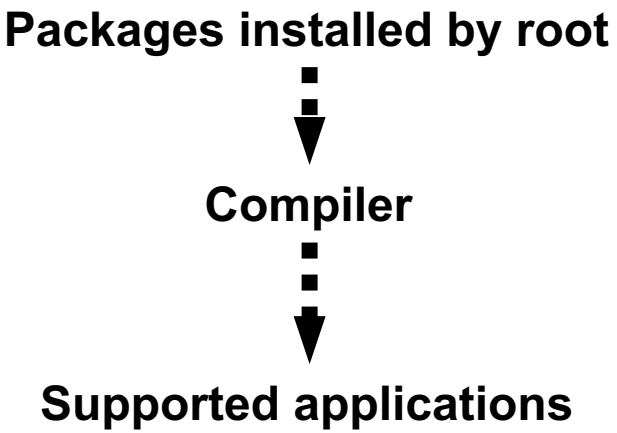
Travis – PROMD demo



CSCS Use Case

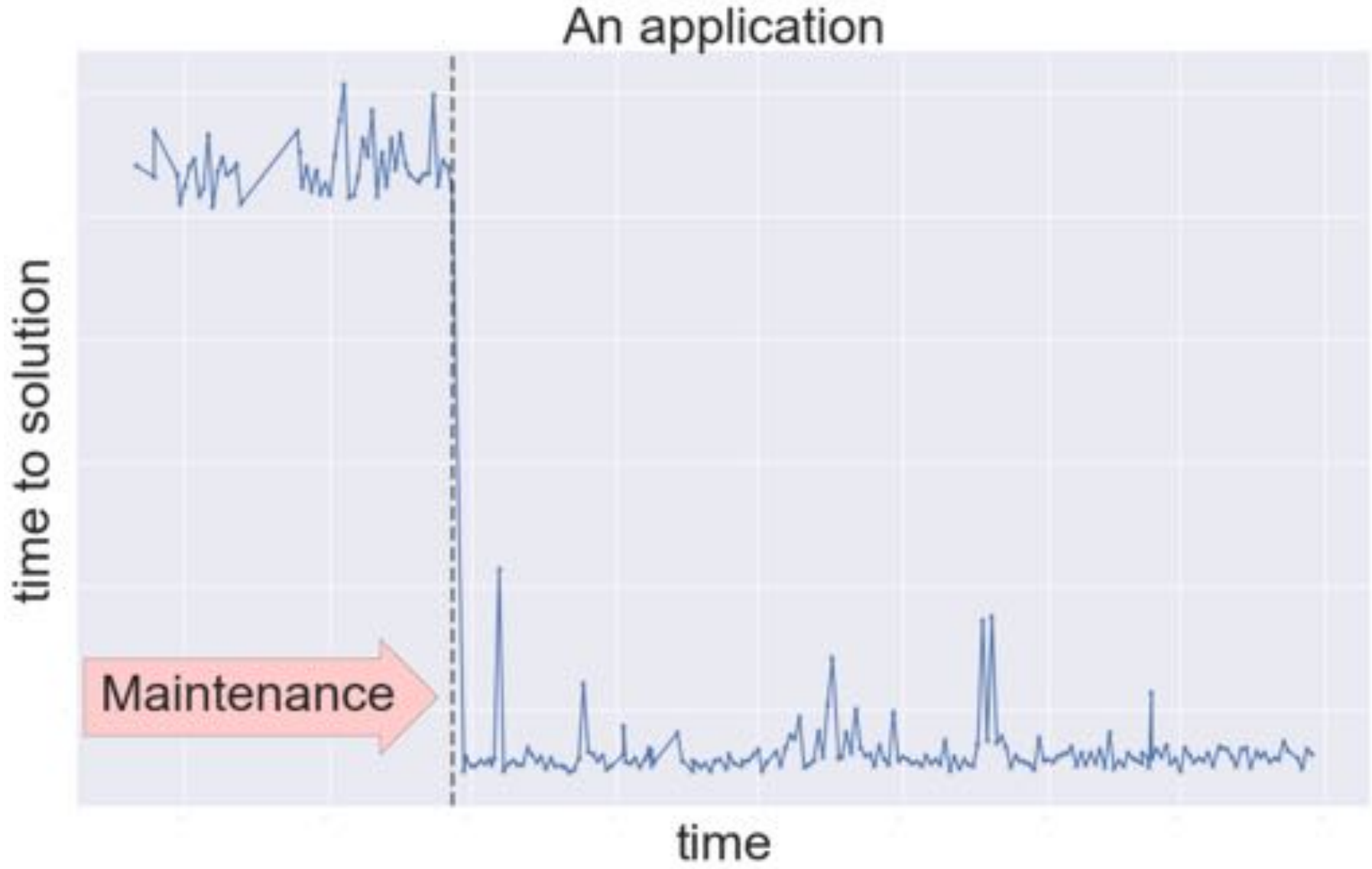
The CSCS Use Case

- ReFrame is used to test all major systems in production
  - The same tests are used for all systems with slight adaptations.
- Wide variety of performance and sanity tests implemented
  - Applications
  - Libraries
  - Programming environment tests
  - I/O benchmarks
  - Performance tools and debuggers
  - Job scheduler tests



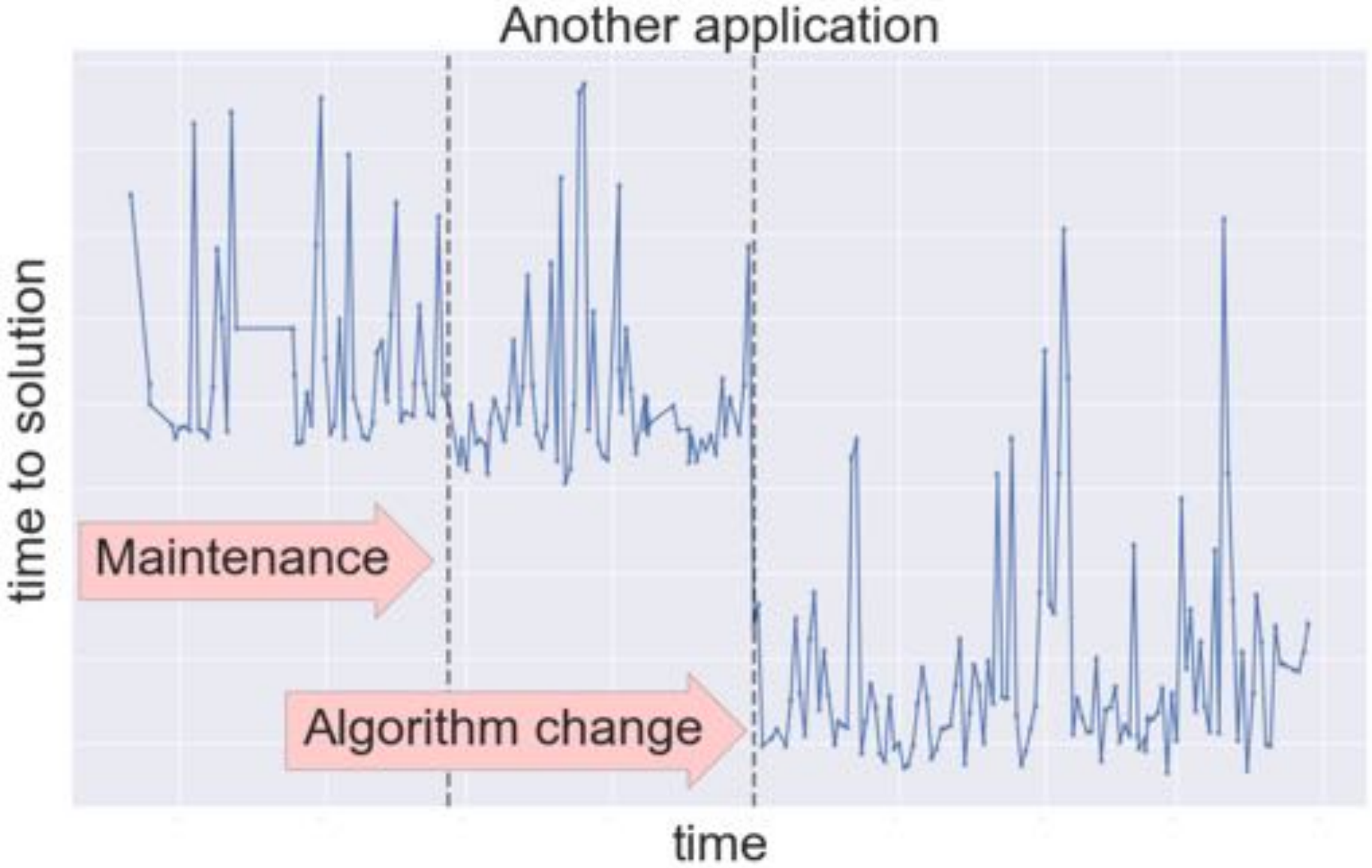
- Two execution modes
  - **Production**: A wide aspect of the sanity and performance tests running daily
  - **Maintenance**: Key functionality and performance tests run during maintenances

The CSCS Use Case



Application optimization

The CSCS Use Case



The CSCS Use Case

Comparison to our former shell script based solution

Maintenance Burden	Shell-script based suite	ReFrame
Total size of tests	14635 loc	2985 loc
Average test file size	179 loc	93 loc
Average effective test size	179 loc	25 loc

5x reduction in the amount of code of regression tests

Open development

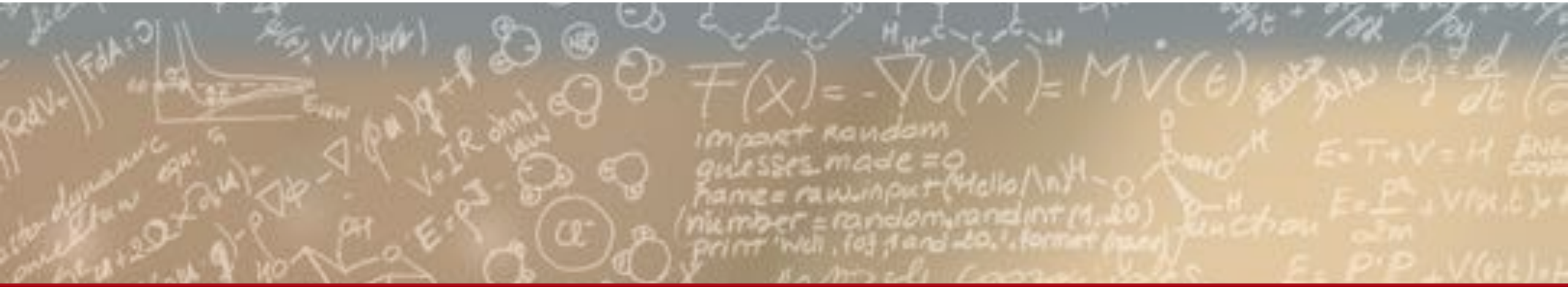
- Development of ReFrame is open on Github
  - https://github.com/eth-cscs/reframe
- Actively developed
  - New features and enhancements are added every month
  - Bugs are addressed promptly
- Quick release cycle (2-3 weeks)
- Hundreds of realistic regression tests
- Full documentation
  - Github.io page: https://eth-cscs.github.io/reframe/index.html
  - Step-by-step tutorial
  - Reference guide

Summary

ReFrame leverages the complexity of regression testing of HPC systems and paves the way for enabling continuous integration of HPC applications.

- *Decouples the logic of the tests from the system details.*
- *Lets you write portable regression tests and decreases their maintenance cost.*
- *Lets you write your regression tests in a modern programming language.*

Try it out, give us some feedback and why not contribute back!



Thank you for your attention.

ReFrame

https://github.com/eth-cscs/reframe