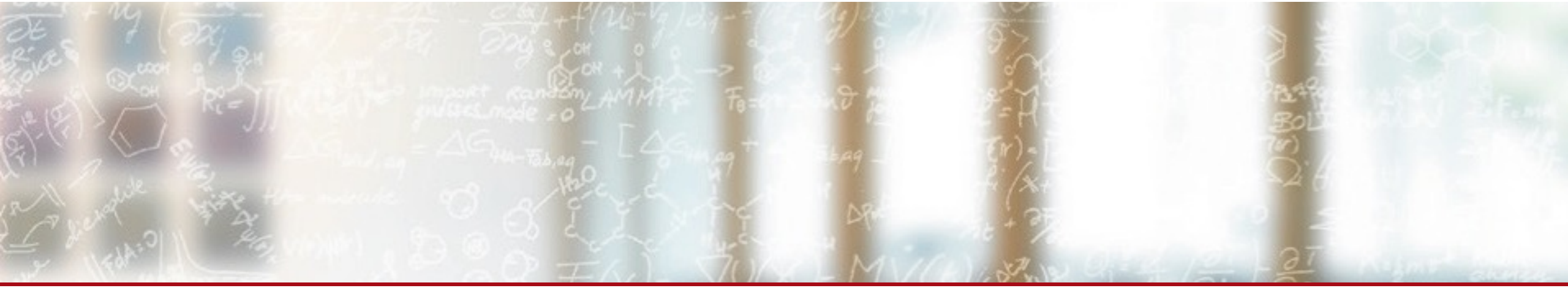
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Continuous validation of software performance at CSCS: How can you contribute?

User Lab Day 2022

Victor Holanda Rusu, CSCS

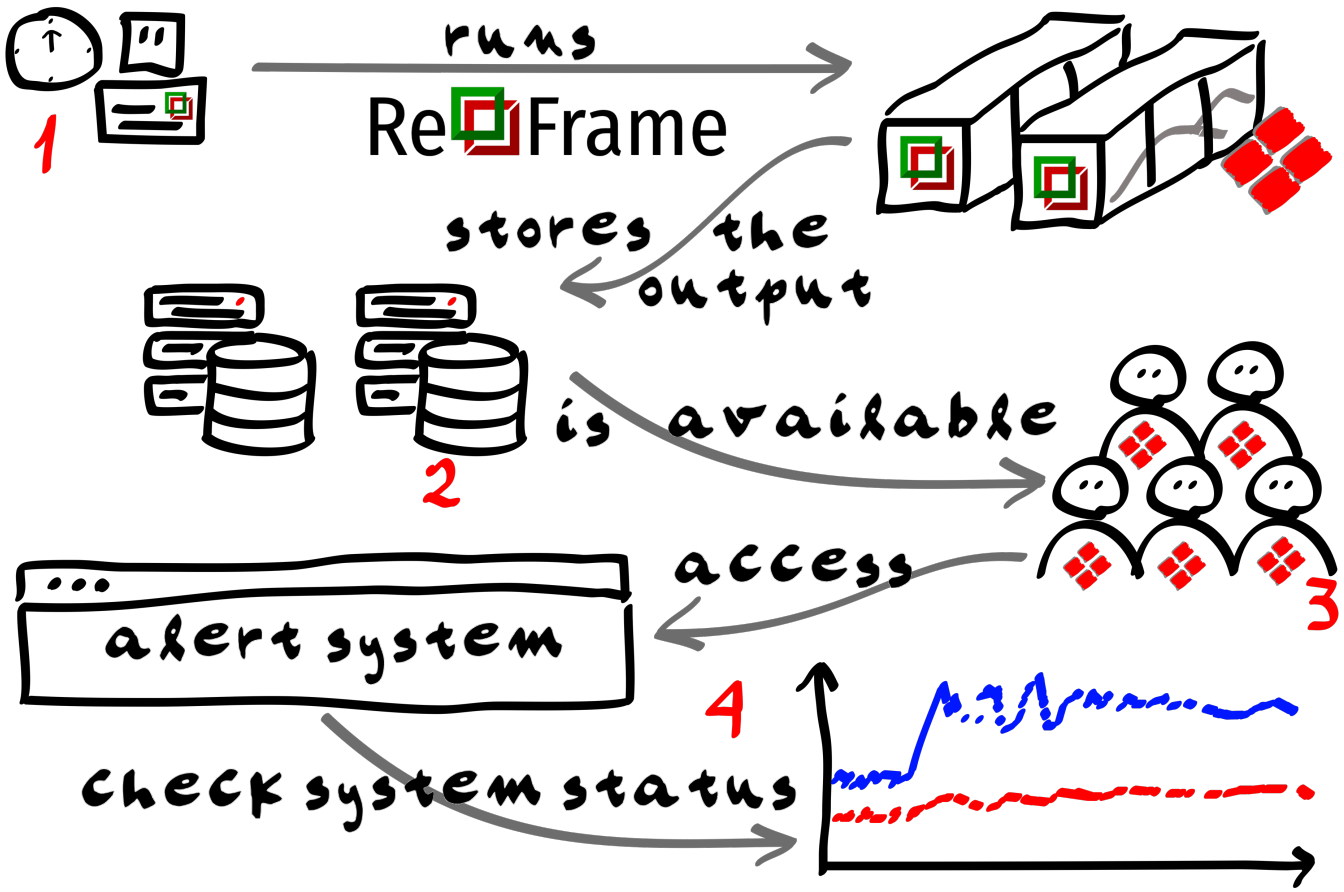
September 2nd, 2022

Regression testing at CSCS

This is all about quality of service

- **Everyday at CSCS**…

1. A robot runs a **selection of ReFrame checks** on **every** User Lab system
2. The **test results are stored** in a dedicated database
3. CSCS **engineers check** the results of the tests
4. And control if there is **any performance degradation** in any system



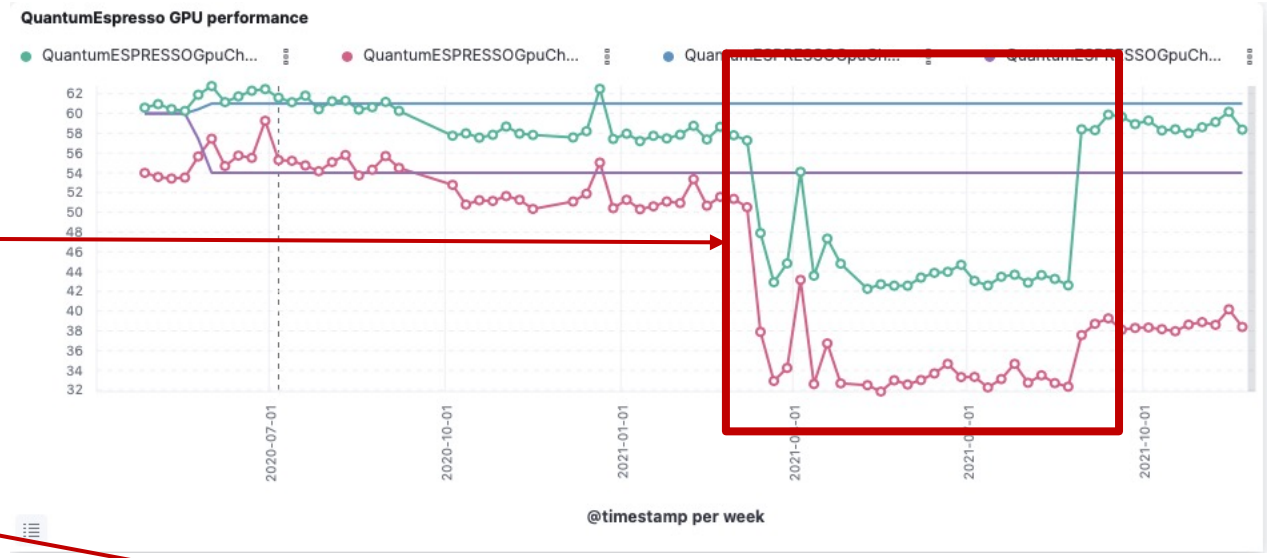
Regression testing at CSCS

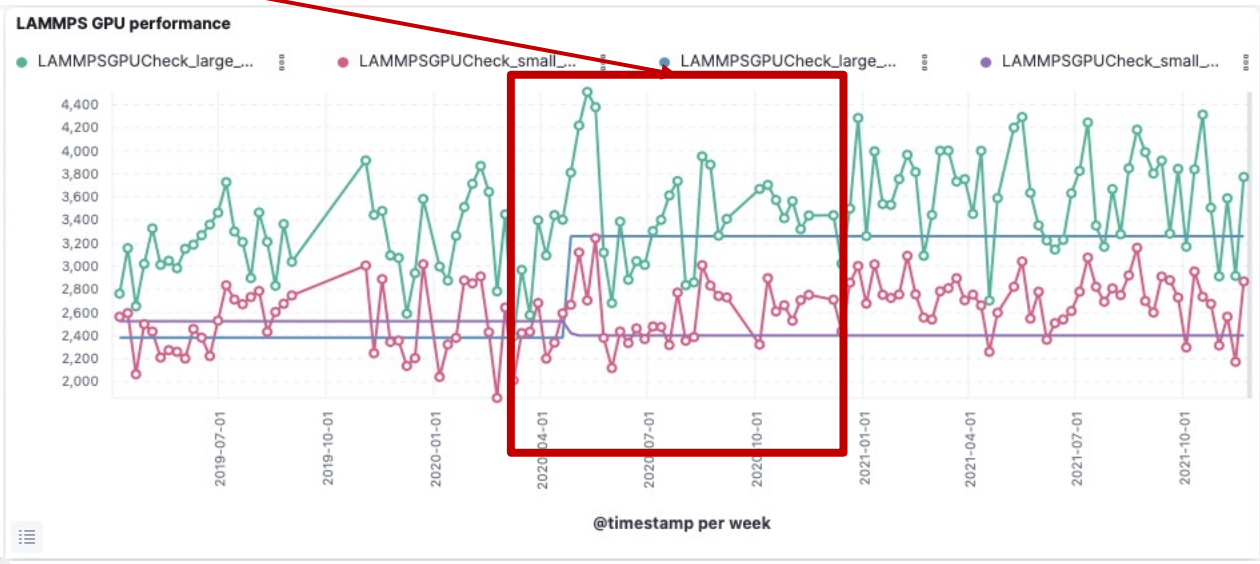
How do the results look like?

Maybe your application behaves like this one?

Results improved upon system change

But no significant changes happened for other metrics

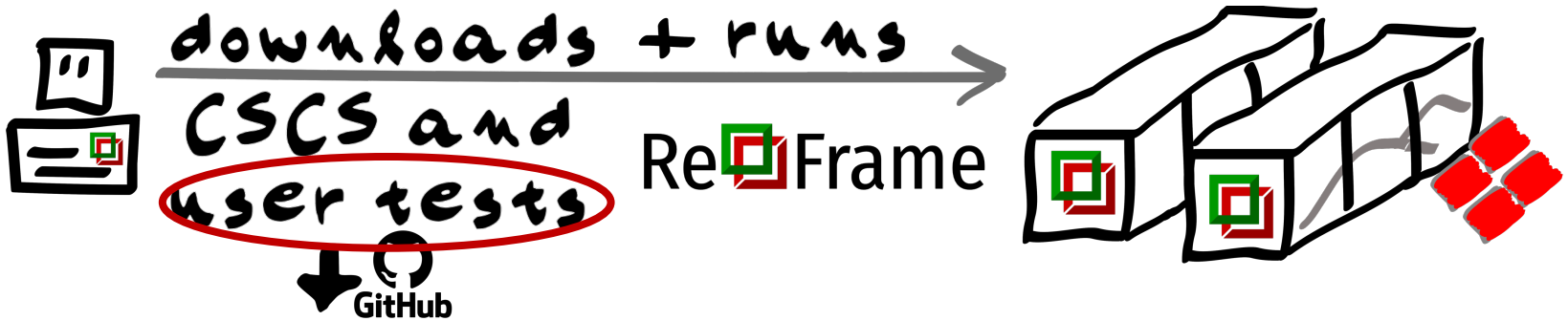




Regression testing at CSCS

Where can you contribute?

- For us it is very important to monitor the system from the user perspective
- On top of system testing and synthetic micro benchmarks
- We also test
  - Tools that users can run
  - Applications and libraries that users can run
- But the day only has 24 hours and we are not experts in every single application running on our systems

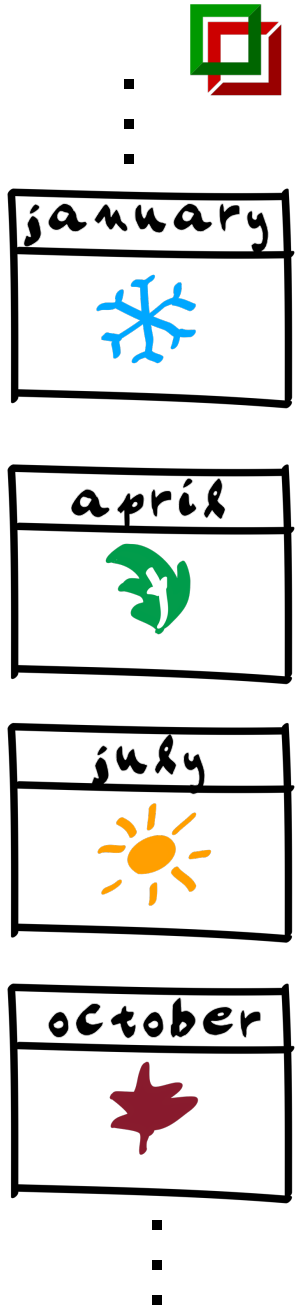


That's where/why we need your help

Contributing to ReFrame

What is there for you?

- Help us identify changes to the **current system** that affect your workflow
- Help us identify changes to the **next system** that affect your workflow
- **Reuse** your tests in **your own lab** environment
- Tests will be **portable** across different HPC sites
- We will be able to provide **better** overall quality of service **to you**
- Tests **can be re-used** for the next project proposal **technical report**



Help us, help you!

Join the ReFrame team

ReFrame at CSCS

What is ReFrame?

Tests pipeline

- ReFrame is an Open Source Regression Testing framework that started at CSCS and has evolved to a community driven project
- It was designed from the ground up for writing portable validation, regression and performance tests running from laptops to Top500 supercomputers
- First public release in May 2017 on Github
  - 42 contributors, 78 forks, 154 stars
  - 70 releases
- Contains several different characteristics, among them
  - Implements a Python eDSL allowing to write tests in a high-level declarative way
  - Tests are composable by design and can be extended or reused across sites
  - Multi-dimensional test parameterisation
  - Support for native and containerised runs
  - Seamless integration with Gitlab CI/CD pipelines

ReFrame – Efficient System and Application Performance Testing

ReFrame tests

What do they look like?

lammps_check.py

```python
import os

import reframe as rfm
import reframe.utility.sanity as sn

class LAMMPSCheck(rfm.RunOnlyRegressionTest):
    scale = parameter(['small', 'large'])
    modules = ['cray-python', 'LAMMPS']
    tags = {'external-resources', 'maintenance', 'production'}
    maintainers = ['LM']
    strict_check = False
    extra_resources = {
        'switches': {
            'num_switches': 1
        }
    }
```

Tests are Python Classes

Parameterise the tests

Define what's the performance metric

Define what's the sanity metric

Make use of modules available at CSCS

We can store the input files for you in case they are very large

```python
    @run_after('init')
    def setup_by_system(self):
        # Reset sources dir relative to the SCS apps prefix
        self.sourcesdir = os.path.join(self.current_system.resourcesdir,
                                       'LAMMPS')
        if self.current_system.name in ['eiger', 'pilatus']:
            self.valid_prog_environs = ['cpeGNU']
        else:
            self.valid_prog_environs = ['builtin']

    @performance_function('timesteps/s')
    def perf(self):
        return sn.extractsingle(r'\s+(?P<perf>\S+) timesteps/s',
                                self.stdout, 'perf', float)

    @sanity_function
    def assert_energy_diff(self):
        energy_reference = -4.6195
        energy = sn.extractsingle(
            r'\s+500000(\s+\S+){3}\s+(?P<energy>\S+)\s+\S+\s\n',
            self.stdout, 'energy', float)
        energy_diff = sn.abs(energy - energy_reference)
        return sn.all([
            sn.assert_found(r'Total wall time:', self.stdout),
            sn.assert_lt(energy_diff, 6e-4)
        ])
```

ReFrame tests

What do they look like?

lammps_check.py

Anotate to make tests runable

Set the executable and its options

```
52  @rfm.simple_test
53  class LAMMPSGPUCheck(LAMMPSCheck):
54      valid_systems = ['daint:gpu']
55      executable = 'lmp_mpi'
56      executable_opts = ['-sf gpu', '-pk gpu 1', '-in in.lj.gpu']
57      variables = {'CRAY_CUDA_MPS': '1'}
58      num_gpus_per_node = 1
59      refs_by_scale = {
60          'small': {
61              'dom:gpu': {'perf': (3456.792, -0.10, None, 'timesteps/s')},
62              'daint:gpu': {'perf': (1566.979, -0.10, None, 'timesteps/s')}
63          },
64          'large': {
65              'daint:gpu': {'perf': (2108.561, -0.10, None, 'timesteps/s')}
66          }
67      }
```

Act upon the different pipeline stages

```
69      @run_after('init')
70      def setup_by_scale(self):
71          self.descr = f'LAMMPS GPU check (version: {self.scale})'
72          if self.scale == 'small':
73              self.valid_systems += ['dom:gpu']
74              self.num_tasks = 12
75              self.num_tasks_per_node = 2
76          else:
77              self.num_tasks = 32
78              self.num_tasks_per_node = 2
79
80          self.reference = self.refs_by_scale[self.scale]
```

Make use of good software practices

```
83  @rfm.simple_test
84  class LAMMPSCPUCheck(LAMMPSCheck):
85      valid_systems = ['daint:mc', 'eiger:mc', 'pilatus:mc']
86      refs_by_scale = {
87          'small': {
88              'dom:mc': {'perf': (4216.05, -0.10, None, 'timesteps/s')},
89              'daint:mc': {'perf': (2523.077, -0.10, None, 'timesteps/s')},
90              'eiger:mc': {'perf': (3807.095, -0.10, None, 'timesteps/s')},
91              'pilatus:mc': {'perf': (4828.986, -0.10, None, 'timesteps/s')}
92          },
93          'large': {
94              'daint:mc': {'perf': (2076.665, -0.10, None, 'timesteps/s')},
95              'eiger:mc': {'perf': (4922.81, -0.10, None, 'timesteps/s')},
96              'pilatus:mc': {'perf': (7247.484, -0.10, None, 'timesteps/s')}
97          }
98      }
```

ReFrame tests

What do they look like?

amber_check.py

```
6  import contextlib
7  import reframe as rfm
8  from hpctestlib.sciapps.amber.nve import amber_nve_check
9
10
11 @rfm.simple_test
12 class cscs_amber_check(amber_nve_check):
13     modules = ['Amber']
14     valid_prog_environs = ['builtin']
15     extra_resources = {
16         'switches': {
17             'num_switches': 1
18         }
19     }
20     tags |= {'maintenance', 'production'}
21     maintainers = ['VH', 'SO']
22     num_nodes = parameter([1, 4, 6, 8, 16], loggable=True)
23     allref = {
24         1: {
25             'p100': {
26                 'Cellulose_production_NVE': (30.84, -0.10, None, 'ns/day'),
```

Tests can be based on the library of tests

Skip tests if conditions are not met for a given system

```
92  @run_after('setup')
93  def skip_if_no_topo(self):
94      proc = self.current_partition.processor
95      pname = self.current_partition.fullname
96      if not proc.info:
97          self.skip(f'no topology information found for partition {pname!r}')
98
99  @run_after('setup')
100 def set_num_tasks(self):
101     if self.variant == 'cuda':
102         self.num_tasks_per_node = 1
103     else:
104         proc = self.current_partition.processor
105         pname = self.current_partition.fullname
106         self.num_tasks_per_node = proc.num_cores
107
108     self.num_tasks = self.num_nodes * self.num_tasks_per_node
```

CPU topology detection can be used to set the number of tasks a test may need

Tests can be written focusing already in strong scaling

How can you contribute?

Make PRs!

- Contribute tests to run at CSCS https://github.com/eth-cscs/cscs-reframe-tests
- Contribute to the library of tests at https://github.com/reframe-hpc/reframe
- Contribute to the framework at https://github.com/reframe-hpc/reframe
- In case of questions open issues at https://github.com/reframe-hpc/reframe

ReFrame

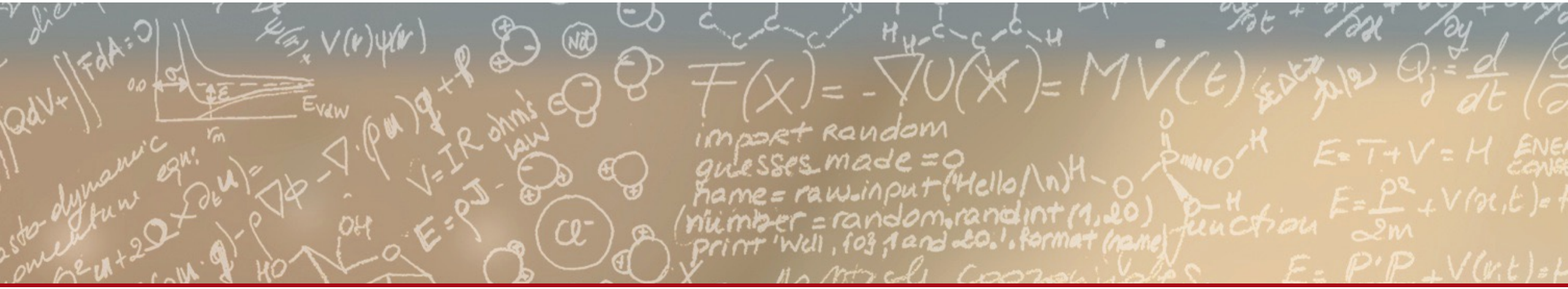
https://reframe-hpc.readthedocs.io

https://github.com/reframe-hpc/reframe

https://reframe-slack.herokuapp.com/

@ReFrameHPC

We are a community and we are here to help!



Thank you for your attention.

Questions?

ReFrame

https://reframe-hpc.readthedocs.io

https://github.com/reframe-hpc/reframe

https://reframe-slack.herokuapp.com/

@ReFrameHPC

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