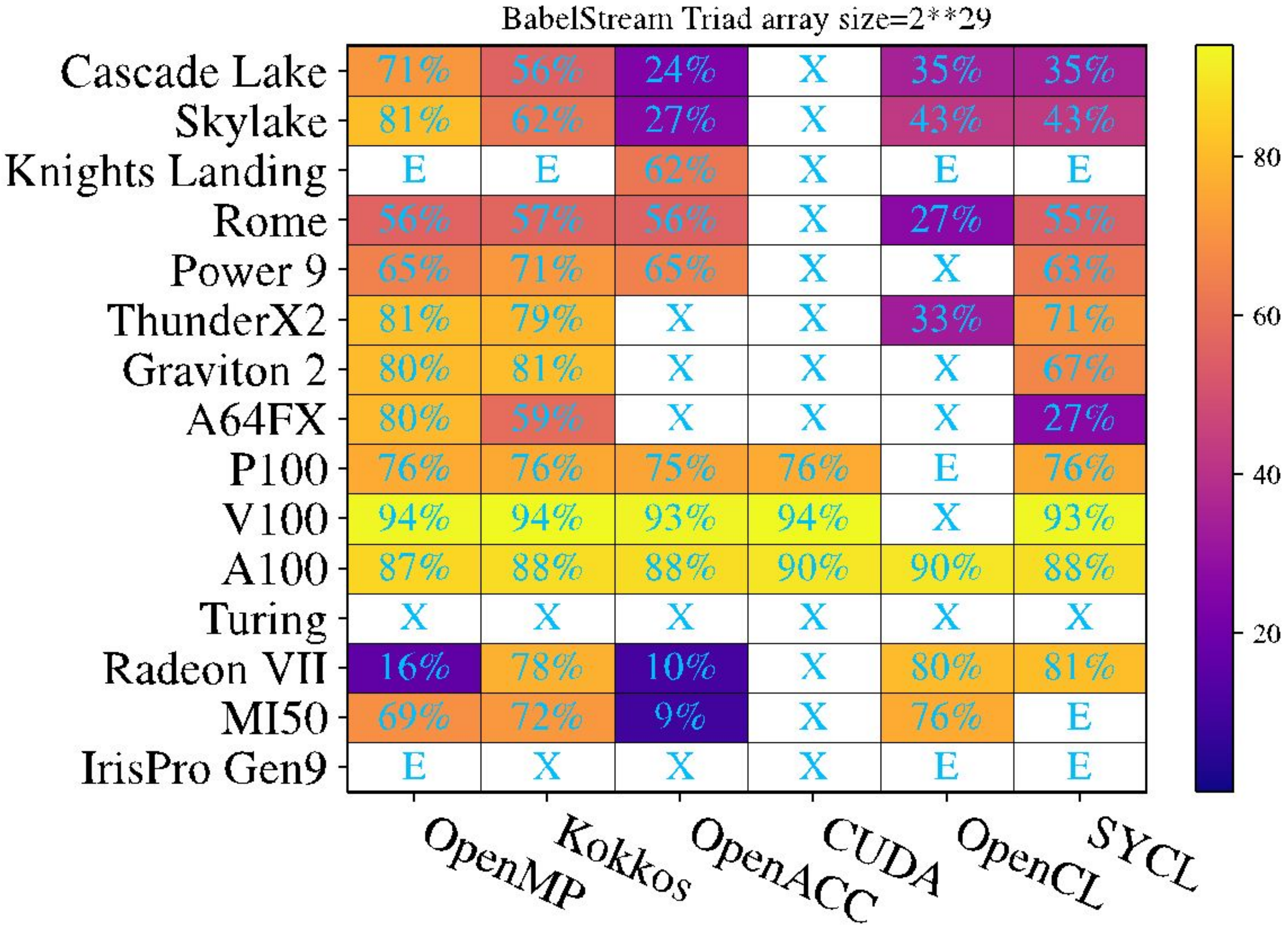
Portable and Reproducible Benchmarking for Exascale Systems

Ilektra Christidi[1], Tuomas Koskela[1], Mose Giordano[1], Emily Dubrovska[1]
Tom Deakin[2], Kaan Olgu[2], Chris Maynard[4], David Case[3], Jamie Quinn[1]

1 - Advanced Research Computing Centre, UCL
2 - Advanced Computer Systems, University of Bristol
3 - Computer Science, University of Reading
4 - Met Office



BabelStream Triad array size=2**29

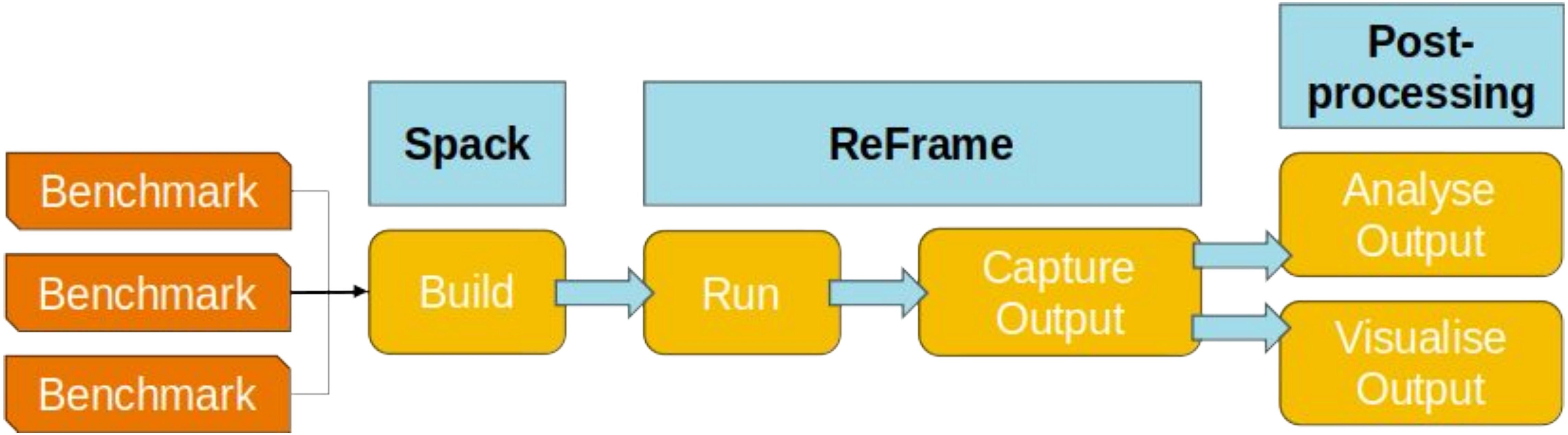
	OpenMP	Kokkos	OpenACC	CUDA	OpenCL	SYCL
Cascade Lake	71%	56%	24%	X	35%	35%
Skylake	81%	62%	27%	X	43%	43%
Knights Landing	E	E	62%	X	E	E
Rome	56%	57%	56%	X	27%	55%
Power 9	65%	71%	65%	X	X	63%
ThunderX2	81%	79%	X	X	33%	71%
Graviton 2	80%	81%	X	X	X	67%
A64FX	80%	59%	X	X	X	27%
P100	76%	76%	75%	76%	E	76%
V100	94%	94%	93%	94%	X	93%
A100	87%	88%	88%	90%	90%	88%
Turing	X	X	X	X	X	X
Radeon VII	16%	78%	10%	X	80%	81%
MI50	69%	72%	9%	X	76%	E
IrisPro Gen9	E	X	X	X	E	E

From https://doi.org/10.1109/P3HPC51967.2020.00006

Summary

- It’s important to understand how our applications perform on different HPC architectures in order to make informed decisions about **future systems** and **future software**
- Benchmarking, as it’s traditionally done, requires a lot of manual effort, is time-consuming, **error prone and difficult to reproduce**
- **Tools exist** in the community to automate the manual effort – recording and documenting in the process.
- We’ve developed a framework using **Spack** and **ReFrame** to automate portable building and running of benchmarks.
- There is an initial price to pay, but the increase in productivity should be **worth it!** More collaborators are welcome.

Building Blocks of a Benchmarking Workflow



ExCALIBUR Portable Benchmarking Framework v1.0.0

Framework for automating builds, runs and data collection of benchmarks across HPC systems

It contains

- ReFrame configuration and Spack environments for UK HPC systems
- Benchmark applications from collaborators
- Analysis and Visualisation tools
- Documentation and Tutorials
- Python packaging

It is **not**

- An authoritative collection of benchmarks
- A benchmarking campaign

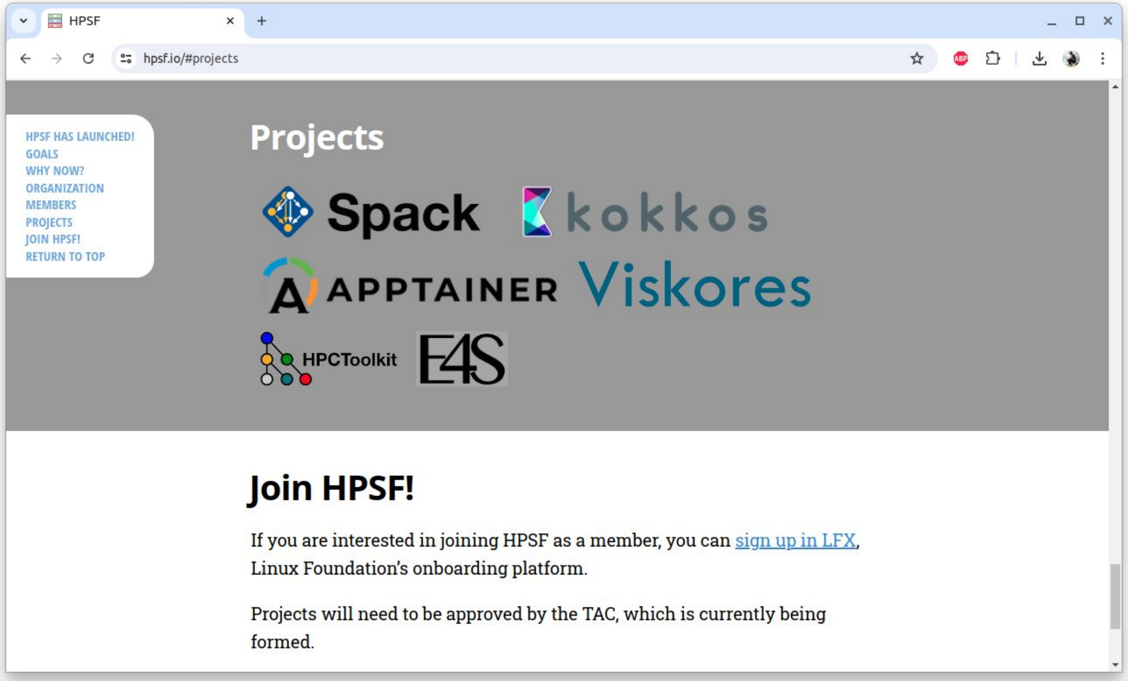
GitHub https://github.com/ukri-excalibur/excalibur-tests

Using Spack for building benchmarks

Spack manages dependencies and automates builds

- Developed in US national labs, used heavily in ECP
- Supports a variety of build systems
- Build steps are recorded
- Maintains a repository of build recipes
- Customizable virtual environments
- Flexible python interface
- Easy to install and run with user permissions

github.com/spack/spack



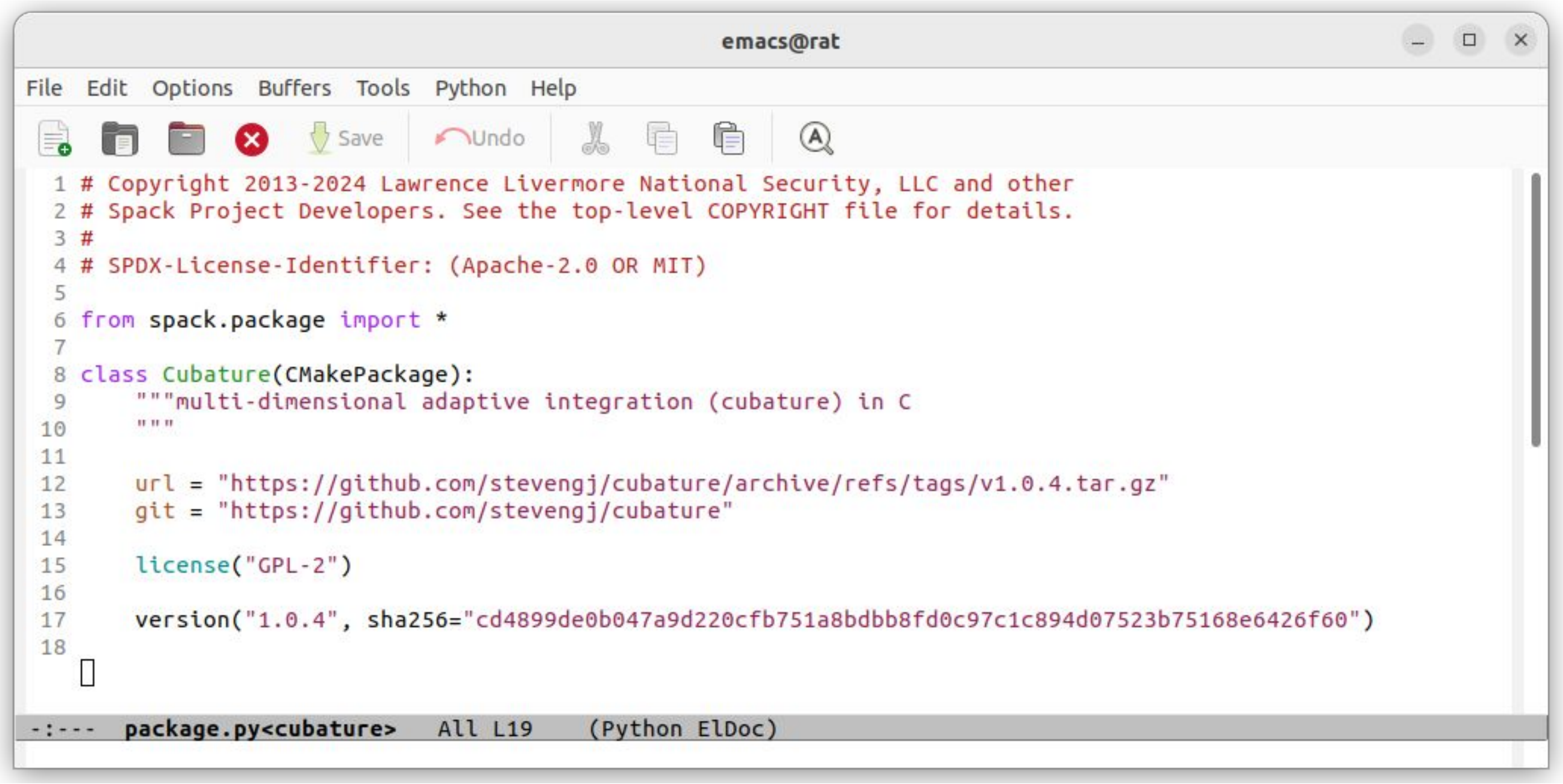
Projects

Join HPSF!

If you are interested in joining HPSF as a member, you can sign up in LFX, Linux Foundation's onboarding platform.

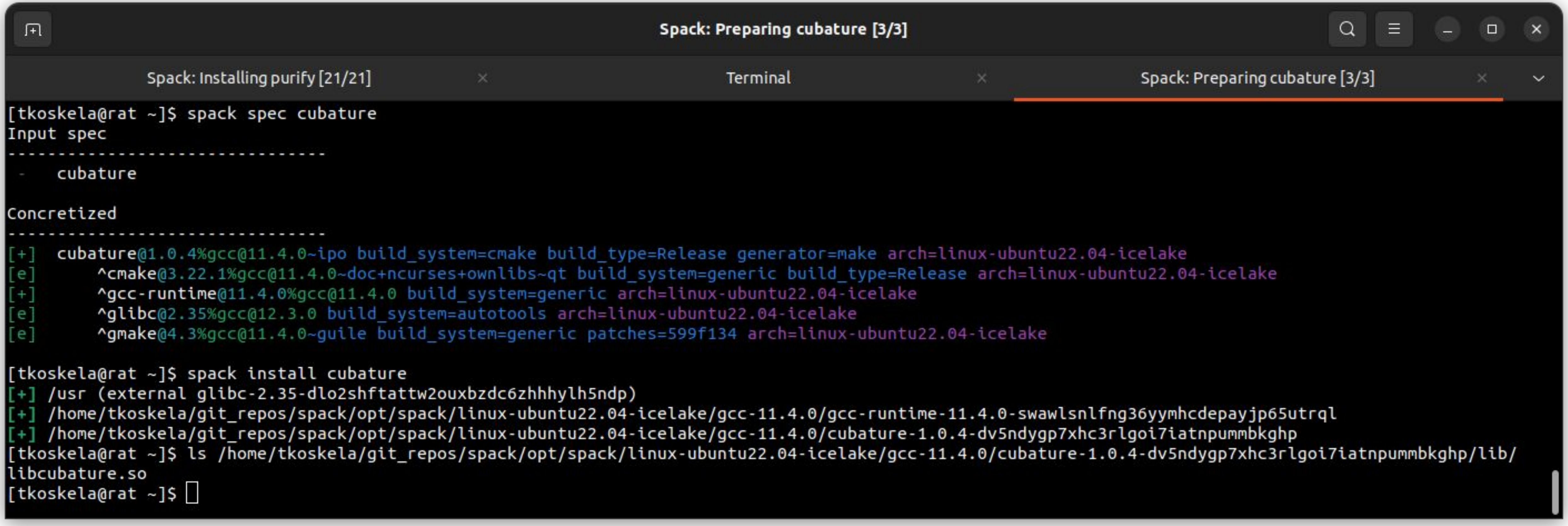
Projects will need to be approved by the TAC, which is currently being formed.

Spack recipe example (1/2)


emacs@rat
File Edit Options Buffers Tools Python Help
Save
Undo
-:--- package.py<cubature>   All L19   (Python ElDoc)

```python
# Copyright 2013-2024 Lawrence Livermore National Security, LLC and other
# Spack Project Developers. See the top-level COPYRIGHT file for details.
#
# SPDX-License-Identifier: (Apache-2.0 OR MIT)

from spack.package import *

class Cubature(CMakePackage):
    """multi-dimensional adaptive integration (cubature) in C
    """

    url = "https://github.com/stevengj/cubature/archive/refs/tags/v1.0.4.tar.gz"
    git = "https://github.com/stevengj/cubature"

    license("GPL-2")

    version("1.0.4", sha256="cd4899de0b047a9d220cfb751a8bdbb8fd0c97c1c894d07523b75168e6426f60")

```

Spack recipe example (2/2)


Spack: Preparing cubature [3/3]
Spack: Installing purify [21/21]
Terminal
Spack: Preparing cubature [3/3]

```
[tkoskela@rat ~]$ spack spec cubature
Input spec
--------------------------------
 -   cubature

Concretized
--------------------------------
[+]  cubature@1.0.4%gcc@11.4.0~ipo build_system=cmake build_type=Release generator=make arch=linux-ubuntu22.04-icelake
[e]      ^cmake@3.22.1%gcc@11.4.0~doc+ncurses+ownlibs~qt build_system=generic build_type=Release arch=linux-ubuntu22.04-icelake
[+]      ^gcc-runtime@11.4.0%gcc@11.4.0 build_system=generic arch=linux-ubuntu22.04-icelake
[e]      ^glibc@2.35%gcc@12.3.0 build_system=autotools arch=linux-ubuntu22.04-icelake
[e]      ^gmake@4.3%gcc@11.4.0~guile build_system=generic patches=599f134 arch=linux-ubuntu22.04-icelake

[tkoskela@rat ~]$ spack install cubature
[+] /usr (external glibc-2.35-dlo2shftattw2ouxbzdc6zhhhylh5ndp)
[+] /home/tkoskela/git_repos/spack/opt/spack/linux-ubuntu22.04-icelake/gcc-11.4.0/gcc-runtime-11.4.0-swawlsnlfng36yymhcdepayjp65utrql
[+] /home/tkoskela/git_repos/spack/opt/spack/linux-ubuntu22.04-icelake/gcc-11.4.0/cubature-1.0.4-dv5ndygp7xhc3rlgoi7iatnpummbkghp
[tkoskela@rat ~]$ ls /home/tkoskela/git_repos/spack/opt/spack/linux-ubuntu22.04-icelake/gcc-11.4.0/cubature-1.0.4-dv5ndygp7xhc3rlgoi7iatnpummbkghp/lib/
libcubature.so
[tkoskela@rat ~]$ 
```

Using ReFrame for running performance benchmarks

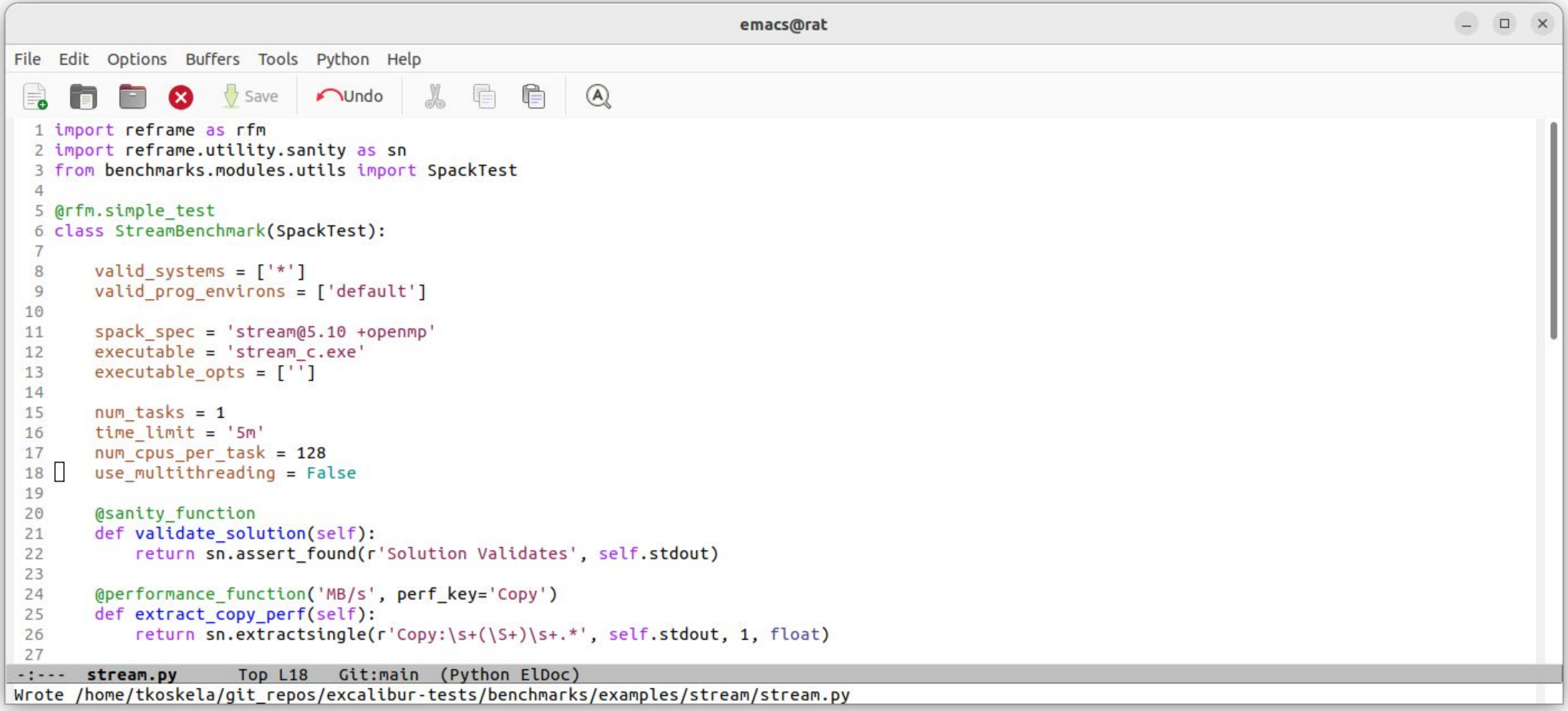
ReFrame

github.com/reframe-hpc/reframe

Framework for regression tests and benchmarks for HPC systems

- From CSCS and ETH Zurich
- Python interface to write benchmarks in a declarative way
- Abstracts interactions with the scheduler and build system
- Tailored for HPC systems
- Supports multidimensional test parametrisation
- Supports many build systems, integrates with spack
- Logs performance, helps keep records

ReFrame class example

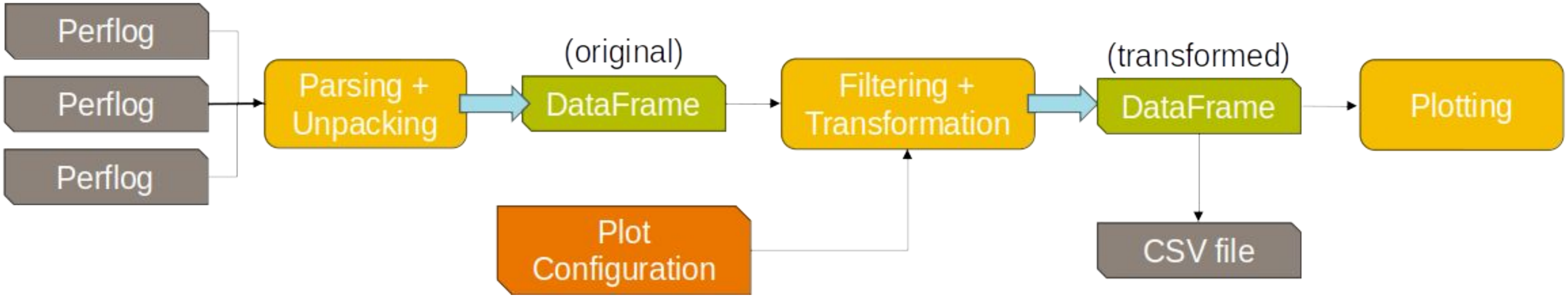


```python
import reframe as rfm
import reframe.utility.sanity as sn
from benchmarks.modules.utils import SpackTest

@rfm.simple_test
class StreamBenchmark(SpackTest):

    valid_systems = ['*']
    valid_prog_environs = ['default']

    spack_spec = 'stream@5.10 +openmp'
    executable = 'stream_c.exe'
    executable_opts = ['']

    num_tasks = 1
    time_limit = '5m'
    num_cpus_per_task = 128
    use_multithreading = False

    @sanity_function
    def validate_solution(self):
        return sn.assert_found(r'Solution Validates', self.stdout)

    @performance_function('MB/s', perf_key='Copy')
    def extract_copy_perf(self):
        return sn.extractsingle(r'Copy:\s+(\S+)\s+.*', self.stdout, 1, float)

```

```
-:---  stream.py      Top L18   Git:main  (Python ElDoc)
Wrote /home/tkoskela/git_repos/excalibur-tests/benchmarks/examples/stream/stream.py
```

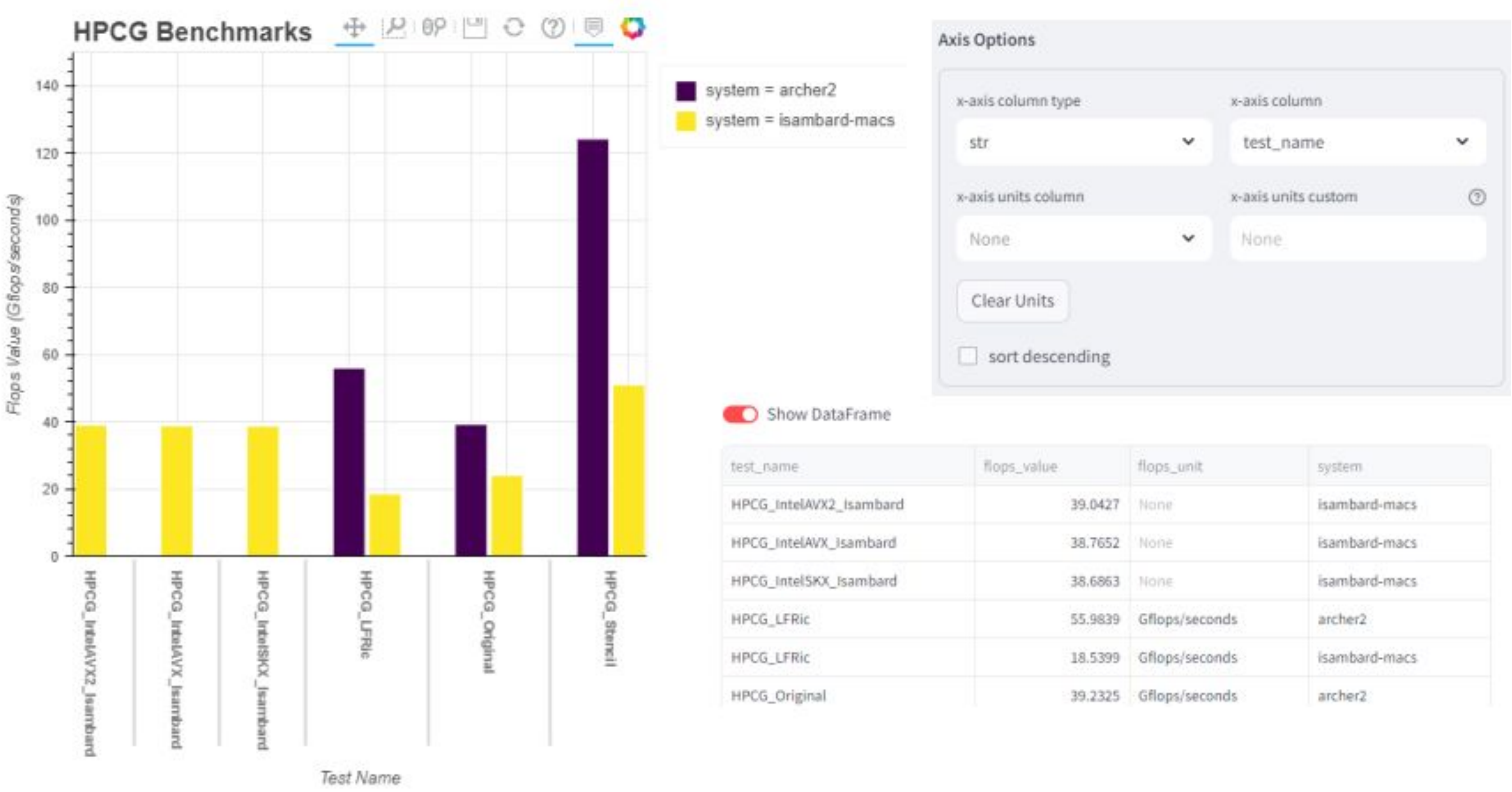
Analysis and Visualisation Workflow

```
python post_processing.py log_path config_path
```



Visualisation GUI

- Post-processing can be run as a web app via **Streamlit**.
- Plot config is now an optional argument because it can be created from scratch in the UI.



Axis Options

x-axis column type: str

x-axis column: test_name

x-axis units column: None

x-axis units custom: None

□ sort descending

Show DataFrame

test_name	flops_value	flops_unit	system
HPCG_IntelAVX2_Isambard	39.0427	None	isambard-macs
HPCG_IntelAVX_Isambard	38.7652	None	isambard-macs
HPCG_IntelSKX_Isambard	38.6863	None	isambard-macs
HPCG_LFRic	55.9839	Gflops/seconds	archer2
HPCG_LFRic	18.5399	Gflops/seconds	isambard-macs
HPCG_Original	39.2325	Gflops/seconds	archer2

Future Plans

- Maintain the suite and support benchmark and system additions.
- Post-processing feature development
- Systematic deployment – include framework as CI on key HPC systems
- Platform for hosting data – curate and store benchmark results

Summary

- It's important to understand how our applications perform on different HPC architectures in order to make informed decisions about **future systems** and **future software**
- Benchmarking, as it's traditionally done, requires a lot of manual effort, is time-consuming, **error prone and difficult to reproduce**
- **Tools exist** in the community to automate the manual effort – recording and documenting in the process.
- We've developed a framework using **Spack** and **ReFrame** to automate portable building and running of benchmarks.
- There is an initial price to pay, but the increase in productivity should be **worth it!** More collaborators are welcome.

Thanks! (Demo)

https://github.com/ukri-excalibur/excalibur-tests