Hewlett Packard
Enterprise

HPE Cray Programming Environment

Testing Overview

Jeff Hudson

May 7, 2023

HPE Cray Programming Environment

Comprehensive set of tools for developing, porting, debugging, and tuning of HPC applications on HPE & HPE Cray systems

Development

Programming Models

- HPE Cray MPI
- SHMEM
- OpenMP | OpenACC (Fortran)
- AMD ROCm HIP | NVIDIA CUDA
- UPC | Fortran co-arrays
- Global Arrays

Programming Environments

- Compiling Environment
- GNU
- Intel Programming Environment
- AMD Programming Environment
- NVIDIA HPC SDK

Programming Languages

- C
- C++
- Fortran
- Python
- R

Optimized Libraries

- LibSci (BLAS)
- LAPACK & ScaLAPACK
- LibSci_ACC
- Iterative Refinementt Toolkit
- FFTW

I/O Libraries

- NetCDF
- HDF5

Debugging

Comparative Debugger
Compare two versions of an application

GDB for HPC
Parallelized gdb for HPC

Valgrind for HPC
Memory debugging at scale

Stack Trace Analysis Tool
Stack tracing at scale

Sanitizers for HPC
Detects memory and thread errors

Tool for Abnormal Termination Processing
Manage core files at scale

TotalView

DDT

Performance Analysis & Optimization

Performance Analysis Tool (PAT)
Whole program performance analysis, exposing wide set of indicators, identifying bottlenecks and automatically generating suggestions to improve performance.

Visualization Tool
Complements text reports with summary of performance data in graphs and charts, allowing users to drill down and resolve issues

Code Parallelization Assistant
Reveal hidden potential of an application via code restructuring

Setup & Runtime

Environment Setup

Tool Enablement
(for Spack, CMake, EasyBuild, etc)

Modules/Lmod

Supported systems:

- HPE Cray EX
- HPE Apollo
- HPE ProLiant DL

HPE—authored | HPE Added-value to 3rd party | 3rd party

HPE Cray Programming Environment

Layers of Testing

Unit

- Automated
- Product Specific Unit level testing
- Written by Developers
- Simple “sanity” Testing
- Some products leverage 3rd party software tests

Functional and Regression

- Automated
- Product focused, but will test product interfaces
- Written by Developer and Testers
- Accounts for OS, ENV, Arch, etc.
- Performed on real HW and virtual environments

Integration

- Automated and Manual mix
- Highly dependent on HW/SW stack
- Performed on real HW

Black Box/App and Performance

- Automated and Manual mix
- Teams analyze benchmark / application metrics
- Verify series of applications for regressions

Unit and Functional/Regression Testing

Testing

- Unit tests run per Commit in build pipelines orchestrated by a Jenkins server
- Number/types of tests vary per CPE product
- Static Code analysis with Linters and Sonar Qube
- Security and vulnerability scanning with Snyk, ClamAV, and in-house scanning tools
- Unit tests limited to single node/instance, while Functional/Regression testing will include multi-node tests.

Combinations

- HPE Cray Products have many architecture, accelerator, OS, and other possible combinations
- Limited testing across these combinations due to hardware/time restrictions
- Separate Builds for
  - OS distro (SLES/COS and RHEL)
  - OS version (SLES SP3, SLES SP4, etc.)
  - Arch (x86 and aarch64)
  - Release, Debug, and other environment combinations

Integration Testing

Testing

- CPE team testing is focused on tool interoperability, user environment, and general compatibility (dependency modeling)
- Mix of automated and manual testing, which includes installing on internal test systems that emulate customer environments
- Other HPE Cray teams, which depend on the Cray Programming Environment, have their own integration test suites (COS, CSM, HPCM, SlingShot, etc.)
- Testing includes 3rd party software integration

Combinations

- Limited to currently supported and some future system configurations due to hardware/time restrictions
- Separate test deployments for different OS and system management environments
- Integration testing of CPE is also done by HPE Cray QA teams (including Recipe testing)

Black Box and Performance Testing

Testing

- Testing run by two teams: Performance and Application Engineering
- Testing leverages a mixture of manual SW installations and automated daily/biweekly installations
- Several engineers and application SMEs support this effort
- Semi-automated regression process with gating focused on high-priority blocker and performance issues
- Example benchmarks and applications would be SPEC, WRF, GROMACS, etc.

Combinations

- Testing mostly limited to representative customer cluster system configurations

Future

Release Management

- Working towards a mechanism to release separate product packages (e.g., Cray Compiler, Perf Tools, Debuggers, etc.)
- Container Releases
- More robust quality metrics and gating processes
- Improved dependency tracking

Testing

- Uniform CI/CD/CT pipelines leveraging GitHub Actions and quality gates
- Better integration of test frameworks like reframe and avocado for use in later stages of CI/CD/CT pipelines
- Provide earlier releases for 3rd party integration testing
- Augment our benchmarking and application level automated testing, targeting customer interests.

Thank You

RESEARCH TECHNOLOGIES
UNIVERSITY INFORMATION TECHNOLOGY SERVICES

Regression Testing at Indiana University with ReFrame

Abhinav Thota

athota@iu.edu

UNIVERSITY INFORMATION
TECHNOLOGY SERVICES

Research Technologies is a division of University Information Technology Services
and a center in the Pervasive Technology Institute at Indiana University.

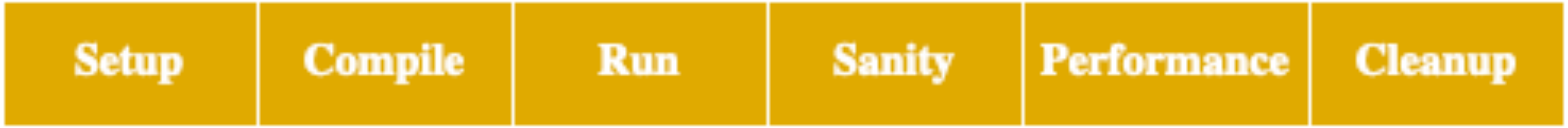
Background

- Regression testing is a routine need, but more important when launching a new system
- Cron based and custom scripted regression testing a decade ago; tried out ReFrame and Pavilion (https://pavilion2.readthedocs.io/en/latest/, LANL) in 2021
- Picked ReFrame based on ease of setting up initial tests and the existence of responsive support through its Slack channel; they are both closely matched and comparable though

ReFrame Highlights

ReFrame

- Uses Python and installs easily (pip install reframe)
- Well documented (https://reframe-hpc.readthedocs.io/en/stable/started.html)
- Has good examples and tutorials (https://reframe-hpc.readthedocs.io/en/stable/tutorial_basics.html)
- Has a user support Slack channel (reframetalk.slack.com) where people are very responsive to questions
- ReFrame workflow:



Setup → Compile → Run → Sanity → Performance → Cleanup

Building Tests

- Many of our tests were created by extending examples that were available online
- Multiple system configurations can be created/supported in the same framework
- ReFrame handles the job submission
  - We can run tests on every node with a single command, for example:

```
reframe -C config.py -c tests/HPLSingleNode.py -J partition=general \
    --system bigred200:general --prgenv='cray$' -r --keep-stage-files \
    --save-log-files --output ./output --distribute="idle" --stage ./stage \
    --report-file ./reports/bigred200_HPL_report_$(date -d "today" +"%Y%m%d%H%M").json \
    --timestamp=%FT-%H-%M
```

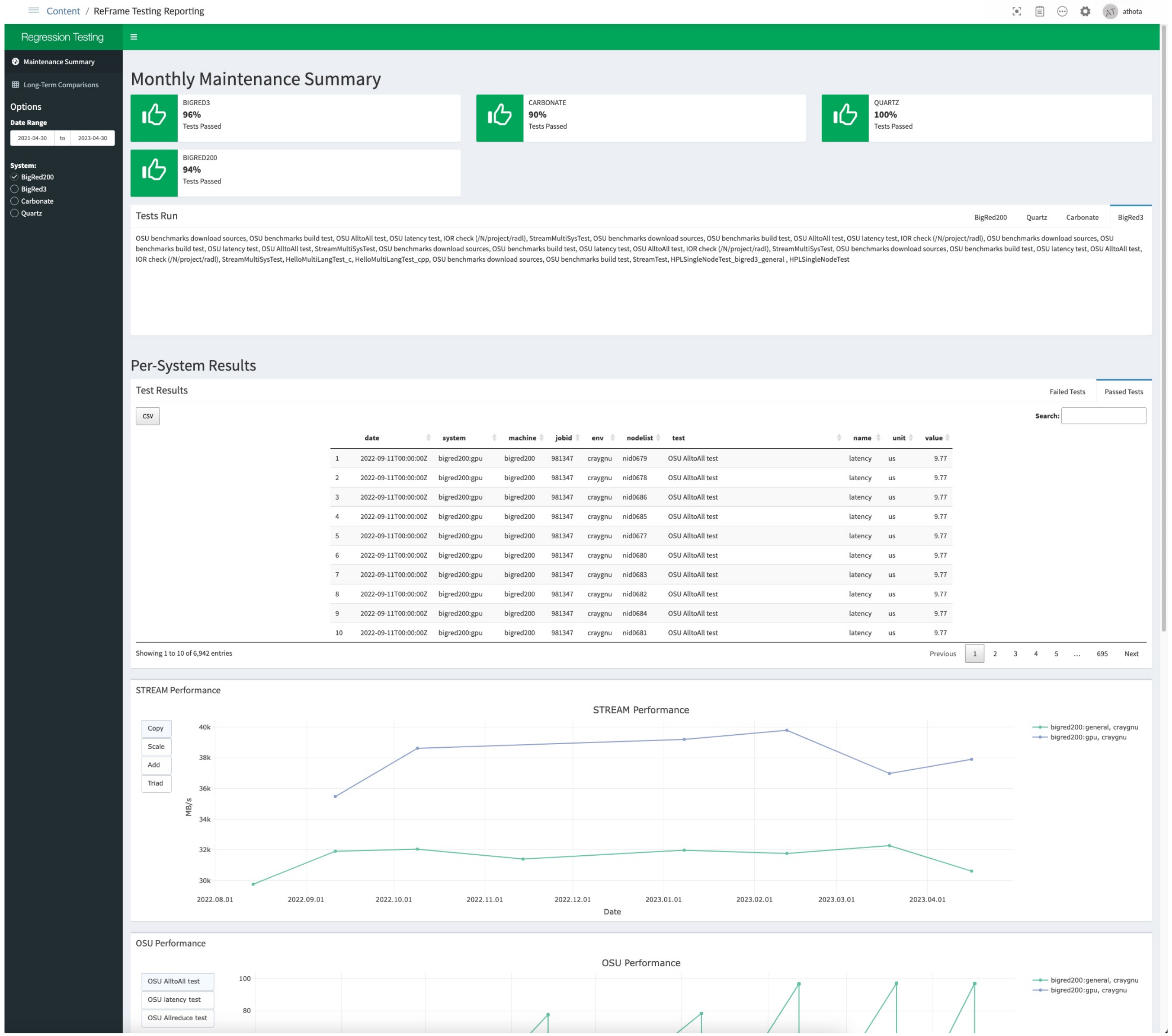
The `–distribute=“idle”` option creates a job for every idle node in Big Red200’s general partition

Learning Curve

- It took at least a couple of weeks to get going
  - There is a lot to understand; often must hunt for the errors because it is not obvious from on screen output
  - Adding new tests took some work initially
- Timeline:
  - Started testing in July 2021
  - Monthly tests starting March of 2022 on Big Red 200 (went into production in April 2022)
- What we are running:
  - OSU Micro Benchmarks (pt2pt latency and collective alltoall)
  - Stream
  - IOR (main Lustre filesystem)
  - HPL (Fall 2022)

Regression Test Dashboard

- We have been publishing the results to an internal PositConnect based dashboard
- Overall, there is a learning curve, but things are smooth now



Options

Date Range

2021-04-30 to 2023-04-30

System:

- ☑ BigRed200
- ○ BigRed3
- ○ Carbonate
- ○ Quartz

Monthly Maintenance Summary

BIGRED3
96%
Tests Passed

CARBONATE
90%
Tests Passed

QUARTZ
100%
Tests Passed

BIGRED200
94%
Tests Passed

Tests Run

OSU benchmarks download sources, OSU benchmarks build test, OSU AlltoAll test, OSU latency test, IOR check (/N/project/radl), StreamMultiSysTest, OSU benchmarks download sources, OSU benchmarks build test, OSU AlltoAll test, OSU latency test, IOR check (/N/project/radl), OSU benchmarks download sources, OSU benchmarks build test, OSU latency test, OSU AlltoAll test, StreamMultiSysTest, OSU benchmarks download sources, OSU benchmarks build test, OSU latency test, OSU AlltoAll test, IOR check (/N/project/radl), StreamMultiSysTest, OSU benchmarks download sources, OSU benchmarks build test, OSU latency test, OSU AlltoAll test, IOR check (/N/project/radl), StreamMultiSysTest, HelloMultiLangTest_c, HelloMultiLangTest_cpp, OSU benchmarks download sources, OSU benchmarks build test, StreamTest, HPLSingleNodeTest_bigred3_general , HPLSingleNodeTest

Per-System Results

Test Results

	date	system	machine	jobid	env	nodelist	test	name	unit	value
1	2022-09-11T00:00:00Z	bigred200:gpu	bigred200	981347	craygnu	nid0679	OSU AlltoAll test	latency	us	9.77
2	2022-09-11T00:00:00Z	bigred200:gpu	bigred200	981347	craygnu	nid0678	OSU AlltoAll test	latency	us	9.77
3	2022-09-11T00:00:00Z	bigred200:gpu	bigred200	981347	craygnu	nid0686	OSU AlltoAll test	latency	us	9.77
4	2022-09-11T00:00:00Z	bigred200:gpu	bigred200	981347	craygnu	nid0685	OSU AlltoAll test	latency	us	9.77
5	2022-09-11T00:00:00Z	bigred200:gpu	bigred200	981347	craygnu	nid0677	OSU AlltoAll test	latency	us	9.77
6	2022-09-11T00:00:00Z	bigred200:gpu	bigred200	981347	craygnu	nid0680	OSU AlltoAll test	latency	us	9.77
7	2022-09-11T00:00:00Z	bigred200:gpu	bigred200	981347	craygnu	nid0683	OSU AlltoAll test	latency	us	9.77
8	2022-09-11T00:00:00Z	bigred200:gpu	bigred200	981347	craygnu	nid0682	OSU AlltoAll test	latency	us	9.77
9	2022-09-11T00:00:00Z	bigred200:gpu	bigred200	981347	craygnu	nid0684	OSU AlltoAll test	latency	us	9.77
10	2022-09-11T00:00:00Z	bigred200:gpu	bigred200	981347	craygnu	nid0681	OSU AlltoAll test	latency	us	9.77

Showing 1 to 10 of 6,942 entries

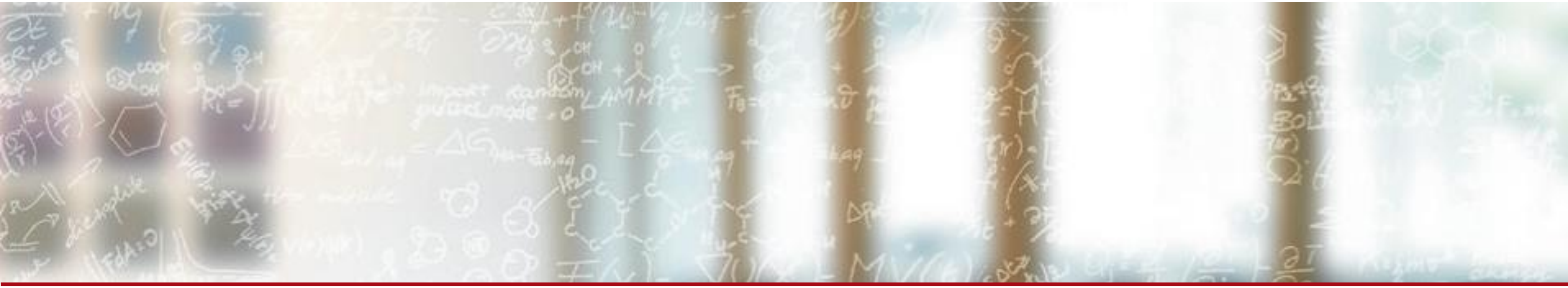
STREAM Performance

OSU Performance

Questions?

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



CSCS Site Presentation

PE Updates and Testing BOF
Eirini Koutsaniti, Guilherme Peretti-Pezzi, CSCS
May 07, 2023

CUG 2023

ReFrame: a python framework for regression and performance testing

- Designed and developed by a small team in CSCS back in 2016
- First public release in May 2017 on Github
  - https://github.com/reframe-hpc/reframe
  - 47 contributors, 88 forks, 176 stars
  - 82 releases
- Used by both academic institutions and industry around the world for testing and benchmarking their clusters
- Key features:
  - High-level tests written in Python
  - Portability across HPC system platforms: support for multiple job schedulers, launchers, build systems, module systems, container engines, etc
  - Comprehensive reports and reproducible methods
  - Easy integration in CI/CD workflows

Future directions and outlook for the framework

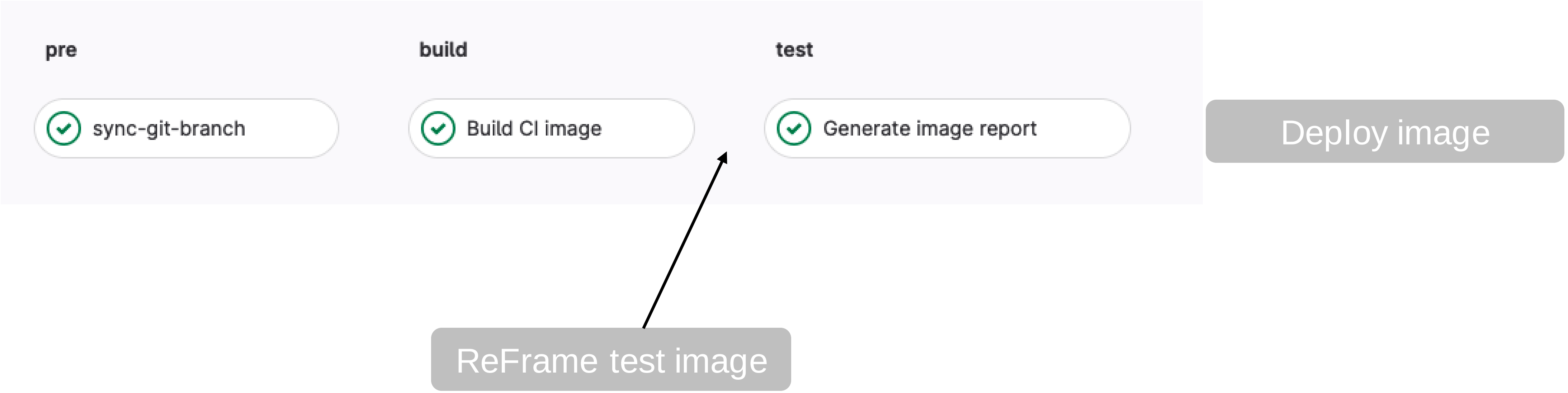
- Encourage collaboration among sites and HPE
  - Expand the test libraries
  - Increase the exchange of tests between sites
  - Explore the prospects of integrating ReFrame in the validation of PEs
  - Explore the possibility of federated CI/CD pipelines with visibility of test results

- Continue improving the framework

Use cases at CSCS

- Production systems currently running ReFrame tests
  - XC
    - Piz Daint (P100 + Broadwell partition)
  - CS-Storm
    - Tsa & Arolla (V100 + Skylake, MeteoSwiss)
  - Shasta
    - Eiger (AMD Rome)
- Other vClusters on Alps (Shasta)
  - Currently 17 vClusters are instantiated
    - Dev & Prod
    - For example
      - Tasna & Balfrin (A100, MeteoSwiss)
      - Clariden (A100 + MI250X)

Future plans: testing at image creation pipeline (GitLab)



pre

✓ sync-git-branch

build

✓ Build CI image

test

✓ Generate image report

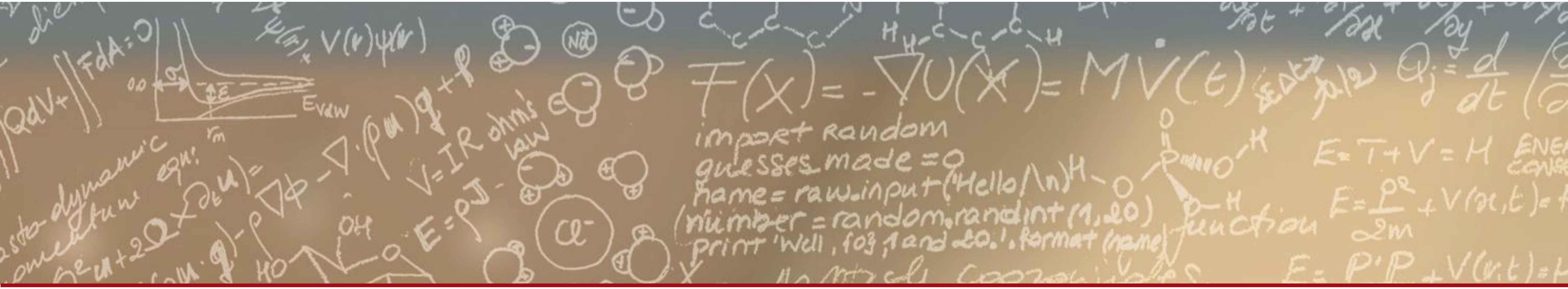
Deploy image

ReFrame test image

CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich


F(x) = -∇U(x) = MV(t)
import random
guesses_made = 0
name = raw_input('Hello\n')
number = random.randint(1, 20)
print 'Well, {0}, I and 20.'.format(name)
function
V=IR ohm's law
E = T+V = H
Evdw

Thank you for your attention.

ReFrame on ARCHER2

Juan F. R. Herrera, EPCC

PEAD, 7th May 2023

www.archer2.ac.uk

Reusing this material

Partners

UKRI

Engineering and Physical Sciences Research Council

Natural Environment Research Council

epcc

THE UNIVERSITY of EDINBURGH

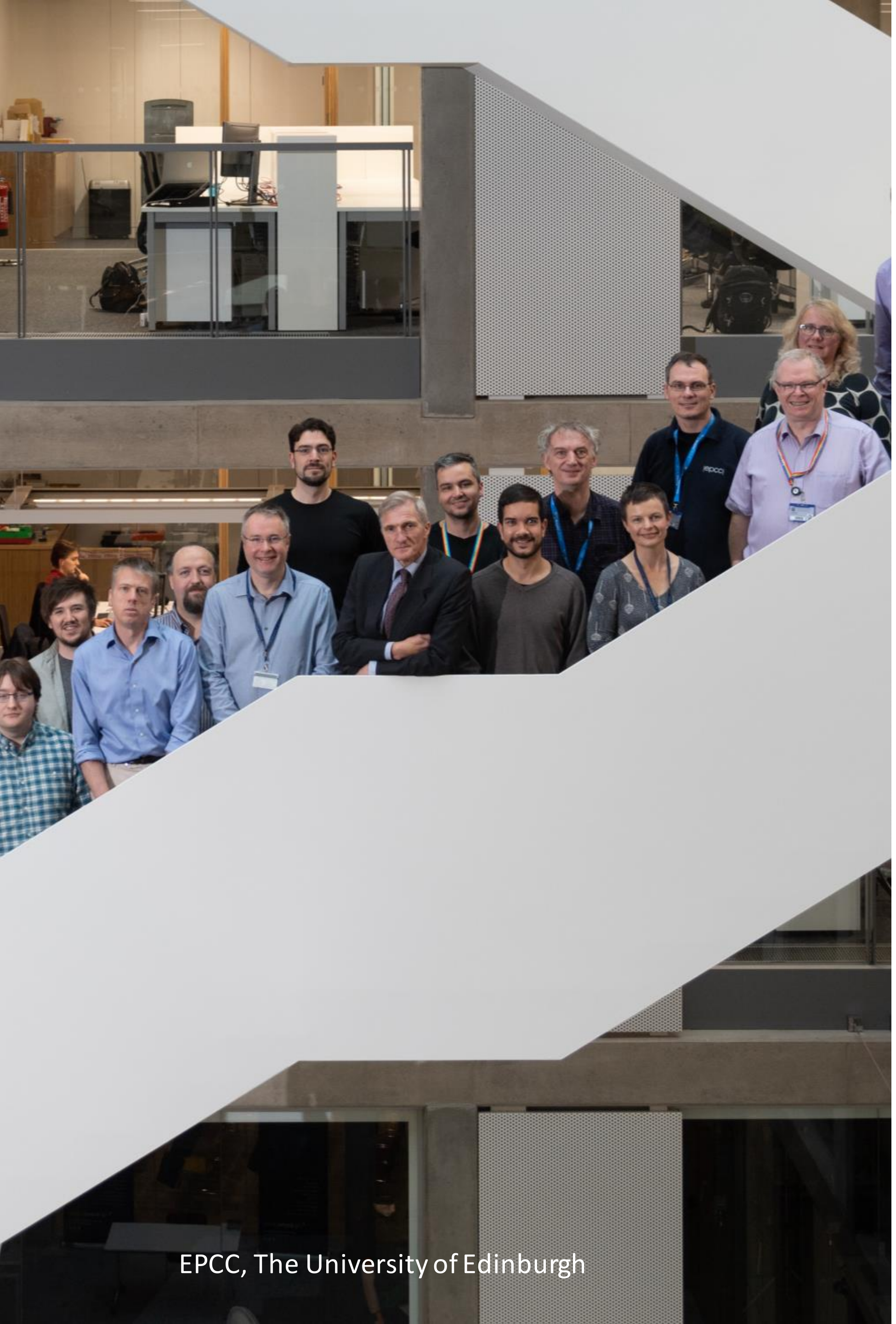
Hewlett Packard Enterprise

ARCHER2

- UK’s National Supercomputing Service
- HPE EX Supercomputer
- 5,860 compute nodes, a total of 750,080 CPU cores
- Compute nodes:
  - Two AMD Zen2 (Rome) EPYC 7742 64-core at 2.25 GHz
  - 256 GB (standard nodes) or 512 GB (high-memory nodes)
  - Two 100 Gbps Slingshot interfaces
- HPE Slingshot 10 interconnect

ARCHER2 storage

- 3x ClusterStor L300 Lustre file systems, each 3.6 PB
- 1 PB ClusterStor E1000F solid state storage
- 4x NetApp FAS8200A file systems, 1 PB total


EPCC, The University of Edinburgh

ARCHER2 Service

- Comprehensive support for users from experts at EPCC and HPE
- Extensive training programme that is free to researchers
    - Wide range of courses from entry level to advanced
- Support to employ Research Software Engineers to improve codes
    - These can be RSEs in the community or provided by EPCC
- Outreach and engagement with the public and wider research community

ReFrame on ARCHER2

- ReFrame is a framework for writing system regression tests and benchmarks, targeted to HPC systems.
- We run the test suite weekly. This allows us to identify any changes in behaviour on the system that may impact users.
- We also want to use this periodic testing to measure performance variability, but this part is still work in progress.
- GitHub repository: https://github.com/EPCCed/epcc-reframe

```
module load reframe
module load epcc-reframe
epcc-reframe -r
```

ReFrame tests

- Login and compute nodes
- Three environments: PrgEnv-cray, PrgEnv-gnu, and PrgEnv-amd
- Tests:
  - Compilation
  - MPI / Interconnectivity: subset of OSU Micro-Benchmarks https://mvapich.cse.ohio-state.edu/benchmarks/
  - Core affinity
  - Apps: CASTEP, CP2K, GROMACS, LAMMPS

ARCHER2 software upgrade

- We are currently undergoing a major software upgrade:
  - Cray OS (COS) 2.0.46 -> 21.04
  - Slingshot interconnect system software (SS) 1.6.0 -> 2.0.2
  - Cray Programming Environment (CPE) 21.04 -> 22.12
- The upgrade has been successfully tested on the ARCHER2 TDS:
  - The ARCHER2 Test and Development System (TDS) is a small system used for testing changes before they are rolled out onto the full ARCHER2 system.
  - 8 compute nodes.
- ReFrame is used to test that the updated system is functional and identify any changes in behaviour of compilers, scheduler, etc.
- Further info: https://docs.archer2.ac.uk/faq/upgrade-2023/

Other HPC systems at EPCC

We plan to use ReFrame on other HPC systems hosted at EPCC such as Cirrus, a Tier-2 system: https://www.cirrus.ac.uk/

Kiitos (thank you)

- Website: https://www.archer2.ac.uk/
- Twitter: @ARCHER2_HPC