How to deal with validation as an HPC software?

An approach to power software testing at scale

Julien Adam — julien.adam@paratools.com

ParaTools

FOSDEM'23
February 5th, 2023

Background

- Back in 2012, from a team developing an MPI runtime
- Shell-script-only regression base
  - Highly dependent on test environments
- Maintenance was a costly task
  - When extending the validation process with new tools
  - When integrating even minor changes from the project

What to expect

- A simple tool
  - Basically, a CLI & minimal configuration files
- Highly customizable tool
  - Test Scenarios may be complex
  - Adaptable to future tech without a complete rewriting
- See Also:
  - ReFrame
  - JuBE
  - Pavilion2

Parallel Computing Validation System

- Shorten as PCVS
- A CLI + YAML-based configuration files
- Testing framework designed to
  - Make test design portable
  - Retarget benchmarks for comparison
  - Autoscale benchmarks to test environments

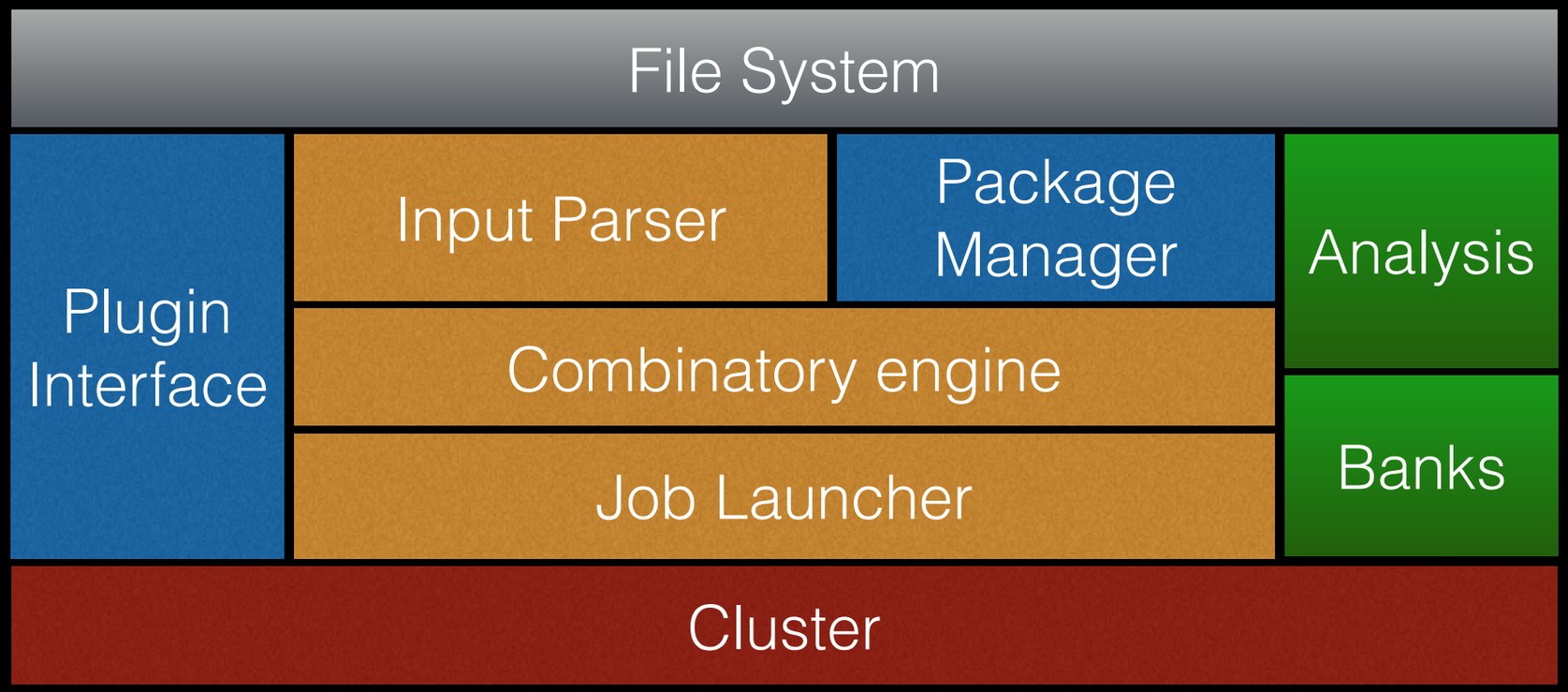
```
 ____                  ____    __   ______                                  __  _
/ __ \____ _____ _____/ / /__  / /  / ____/___ ____ ___ ____  __  __/ /_()___  ____ _
/ /_/ / __ `/ ___/ __ `/ / / _ \/ /  / /   / __ \/ __ `__ \/ __ \/ / / / __/ / __ \/ __ `/
/ ____/ /_/ / /  / /_/ / / /  __/ /  / /___/ /_/ / / / / / / / /_/ / /_/ / /_/ / /_/ / /_/ /
/_/    \__,_/_/   \__,_/_/_/\___/_/   \____/\____/_/ /_/ /_/ .___/\__,_/\__/_/_/ /_/\__,_/
                                                          /_/                     /____/
                                  ⋆ (PCVS) ⋆

 _   __   ___   __   __  _                  ____                  __
| | / /__ _/ (_)__/ /__ _/ /_(_)__  ___    / __/__ _____ _____/ /____ ____ ___
| |/ / _ `/ / / _  / _ `/ __/ / _ \/ _ \   \ \/ / / / ___/ _ \/ _ \/ __/ __ \
| |/ / /_/ / / / /_/ / /_/ / /_/ / / / /   ___/ / /_/ /__  / /_/  __/ / / / /
|___/\__,_/_/_/\__,_/\__,_/\__/_/\___/_/ /_/  /____/\__, /____/\__/\___/_/ /_/ /_/
                                                     /____/
```

Features

- Split environment & test design
  - Test specifications are carried with projects/benchmarks
  - Environment are stored on clusters
- Adapt tests to new environments
  - Auto-retargeting tests to compilers/runtimes
  - Auto-scaling tests to exec environment
- Integrate tools for in-place reporting
- Stand-alone execution (=sessions)
- Simple format: YAML
- Store runs over time
- Run analysis to create trend/stats over time

PCVS Architecture

- Job descriptions expose resource requirements
- Environment provides resources
- => the intersection constitutes the combinatory matrix
- Test workload depends on both these information



PCVS Architecture

- Job descriptions expose resource requirements
- Environment provides resources
- => the intersection constitutes the combinatory matrix
- Test workload depends on both these information

Environment

```yaml
compiler:
  commands:
    cc: mpicc
criterion:
  iterators:
    n_mpi:
      values: [1, 2, 3, 4]
runtime:
  iterators:
    n_mpi:
      option: '-np '
  program: mpirun
```

Job

```yaml
node:
  run:
    program: "./prog"
```

```
mpirun -np 1 <prog>
mpirun -np 2 <prog>
mpirun -np 3 <prog>
mpirun -np 4 <prog>
```

How to write a compilation test

- Alongside with tests/benchmarks
- Static file: `pcvs.yml`
- Multiple build systems supported

Compilation test →
Lang. Autodetect →

```yaml
program_description:
  tag: ["MPI"]
  build:
    files: "*.c" # relative to YML file
    sources:
      binary: "myprogram"
    depends_on: ["deps/program_description"]
    cflags: "extra cflags"
    ldflags: "extra ldflags"
    cwd: "dir/to/build"
    variants:
      - openmp
```

```yaml
autotools:
  params: ['--disable-bootstrap']
cmake:
  vars: ['CMAKE_VERBOSE_MAKEFILE=ON']
make:
  target: all
```

How to write a run test

- Alongside with tests/benchmarks
- Static file: `pcvs.yml`
- Many validation triggers

```yaml
program_description:
  tag: ["MPI"]
  run:
    program: "./a.out"
    iterate:
      n_mpi:
        values: [2, 4]
      program:
        arg:
          option: "-iter"
          values: [10, 100, 1000]
```

Alter global scenario → `n_mpi`

Create your own pattern → `program`

How to write a run test

- Alongsi
- Static f
- Many v

```yaml
validate:
  expect_exit: 0
  time:
    mean: 10.0
    tolerance: 2.0
    kill_after: 20
  match:
    label:
      expr: '^\d+(\.\d+) received$'
      expect: true|false
    label2: 'Total Elapsed: \d+\.\d+ sec.$'
analysis:
  method: "<method>"
script:
  path: "/path/to/script"
```

Alter global sc

Create your ow

```yaml
option: "-iter"
values: [10, 100, 1000]
```

Now Run!

```
$ pcvs profile create mympi --base mpi
$ pcvs run --profile mympi ./MPI/IMB/check
```

- PCVS relies on a single output directory (defaults to **`$PWD/.pcvs-build`**) containing:
  - Configurations
  - Build artefacts (programs, temp files…)
  - PCVS cache (pre-compiled python files)
  - **JSON-formatted Results**
  - Previous results run in the same directory (compressed)

```
MPI/
├── BULL
├── IMB
├── Intel_ANL
├── mpich-3.4.2
├── MPI-IO
├── NAS
├── NBC
├── rma-mt
├── simple
├── Thread_based
└── Threading
OpenMP/
├── BOTS
├── CLOMP
├── EPCC
├── GOMP
├── NAS
├── OpenUH
└── simple
applications/Corals/
├── AMG2013
├── graph500-2.1.4
├── lulesh-2.0.3
├── mcb-20130723
├── miniFe
├── nekbone-2.3.4
└── UMT2013
```

Now Run!

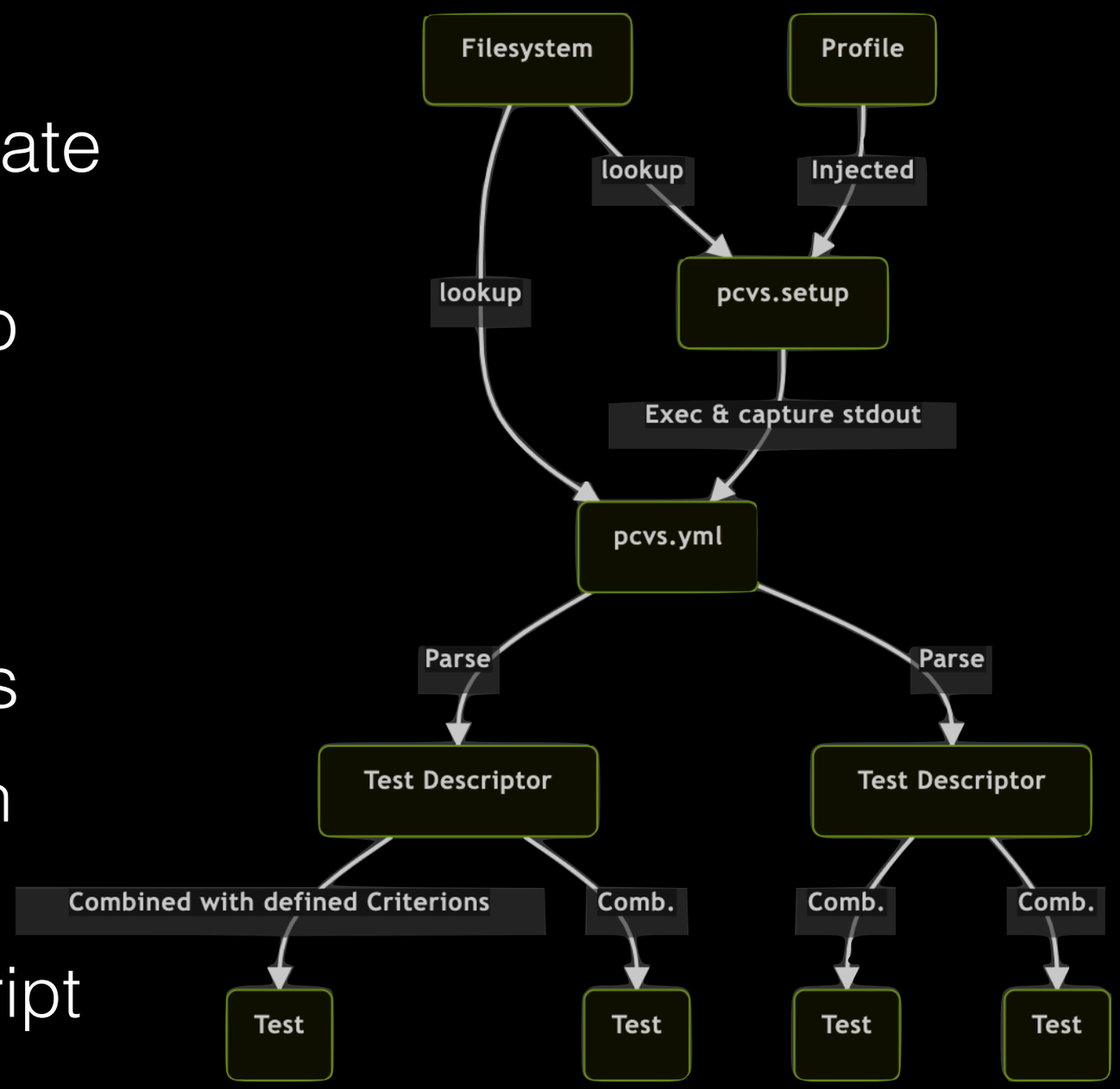
```
Parallel Computing -- Validation System
Copyright © 2017 -- CEA

──────────────────────────────────────── INITIALIZATION ────────────────────────────────────────
◆ Prepare environment
  ◇ Check whether build directory is valid
  ◇ Ensure user-defined programs exist
  ◇ Init & expand criterions
  ◇ Init the global Orchestrator
  ◇ Save Configurations into
◆ Load Test Suites
  ◇ Locate benchmarks from user directories
  ◇ Extract tests from dynamic definitions (1 found)
  ◇ Extract tests from static definitions (0 found)
◆ ===> Processing done in 2.163 sec(s)
──────────────────────────────────────────── SUMMARY ────────────────────────────────────────────
◆ Global Information
  ◇ Date of execution: Tue Jan 31 08:56:29 2023
  ◇ Run by: Julien Adam <adamj@paratools.com>
  ◇ Active session ID: 19
  ◇ Loaded profile: 'user.openmpi'
  ◇ Build stored to:
  ◇ Criterion matrix size per job: 4
  ◇ Bank Management: mpi_repo@openmpi
◆ User directories:
  ◇ IMB:
◆ Globally loaded plugins:
  ◇ TEST_EVAL: MPIPlugin
  ◇ TEST_RESULT_EVAL: BankValidationPlugin
◆ Orchestration infos
  ◇ Test count: 293
  ◇ Max simultaneous Sets: 10
  ◇ Resource count: 10
──────────────────────────────────────────── EXECUTION ──────────────────────────────────────────

  Name                                          SUCCESS   FAILURE   ERROR   OTHER
  ────────────────────────────────────────────────────────────────────────────────
  IMB                                               205        88       0       0

0:14:54 Progress ──────────────────────────────────────────────────────────────────── 100.0%
```

- PCVS can be executed in the background, detached from the current shell.
- Test results are packed when displayed for compactness
- I/O are managed thanks to `rich`

```
$ pcvs exec --list
$ pcvs exec IMB/MPI_Barrier_n2 [--show]
```

Goal: preserve benchmarks

- To ease the benchmark update process, no modifications should be applied directly to sources.
- => Prepare/generate dynamically job descriptions
  - Run a script loading the run environment
  - Output the actual YAML script
  - `pcvs.setup`



Now Report!

- Need: review the results to get direct feedback
  - Ex: Quick rerun
- Tests may be uniquely rerun in the same condition
  - `$ pcvs exec IMB/MPI_Barrier_n2`
- Tests may be GUI-reviewed (Flask Server)
  - From any build or generated archive
  - `$ pcvs report`

Now Report!

- Need: review the results to get direct feedback
  - Ex: Quick rerun
- Tests may be uniquely rerun in the same condition
  - `$ pcvs exec IMB/MPI_Barrier_n2`
- Tests may be GUI-reviewed (Flask Server)
  - From any build or generated archive
  - `$ pcvs report`

Progress	Name	Test Count
	IMB	293

	Name	status	Elapsed time (s)
+	IMB/Ibarrier_args=Ibarrier_n1	SUCCESS	3.36
+	IMB/Unidir_Get_args=Unidir_Get_n1	SUCCESS	3.40
+	IMB/Ibarrier_args=Ibarrier_n2	SUCCESS	3.55
+	IMB/Open_Close_args=Open_Close_n2	SUCCESS	3.67
+	IMB/S_Read_Indv_args=S_Read_Indv_n1	SUCCESS	3.71
+	IMB/P_Read_Priv_args=P_Read_Priv_n1	SUCCESS	3.72
+	IMB/P_Read_Expl_args=P_Read_Expl_n1	SUCCESS	3.75
+	**IMB/S_IWrite_Expl_args=S_IWrite_Expl_n1**	**FAILURE**	**3.75**
+	**IMB/C_Write_Expl_args=C_Write_Expl_n1**	**FAILURE**	**3.77**
+	**IMB/S_Write_Expl_args=S_Write_Expl_n2**	**FAILURE**	**3.80**
+	**IMB/P_IWrite_Priv_args=P_IWrite_Priv_n1**	**FAILURE**	**3.83**
+	IMB/S_Read_Expl_args=S_Read_Expl_n1	SUCCESS	3.83
+	IMB/S_Read_Expl_args=S_Read_Expl_n2	SUCCESS	3.97
+	IMB/S_Read_Indv_args=S_Read_Indv_n2	SUCCESS	4.05
+	IMB/Scatterv_args=Scatterv_n1	SUCCESS	4.07

−	IMB/S_IWrite_Expl_args=S_IWrite_Expl_n4	FAILURE	5.42

```
ccc_mprun -p rome  -n 4                          /pcvs-benchmarks/.pcvs-build/test_suite/IMB/IMB-IO
S_IWrite_Expl
```

```
#---------------------------------------------------------------
#    Intel (R) MPI Benchmarks 2017, MPI-IO part
#---------------------------------------------------------------
# Date                  : Tue Jan 31 09:00:38 2023
# Machine               : x86_64
# System                : Linux
# Release               : 4.18.0-305.62.1.el8_4.x86_64
# Version               : #1 SMP Thu Aug 11 12:07:27 EDT 2022
# MPI Version           : 3.1
# MPI Thread Environment:


# Calling sequence was:

#                              /pcvs-benchmarks/.pcvs-build/test_suite/IMB/IMB-IO S_IWrite_Expl

# Minimum io portion in bytes:   0
# Maximum io portion in bytes:   16777216
#
#
#

# List of Benchmarks to run:

# S_IWrite_Expl


# For nonblocking benchmarks:

# Function CPU_Exploit obtains an undisturbed
# performance of 2071.55 MFlops

#-----------------------------------------------------------------------------------------
```

Setup configuration for your site

- A profile, YAML-based, carries information relative to environment.
  - Nodes, resources, parameters…
- These profiles are stored under a scope for easy access (« a la Git »: global, user & local)
- Base profiles for common use cases are available
- Profiles may even be split up: Config blocks, composable from multiple environments sharing the same filesystem

```yaml
compiler:
  commands:
    cc: mpicc,
    cxx: mpicxx,
    fc: mpif90
criterion:
  iterators:
    n_mpi:
      subtitle: n
      values: [1, 2, 3, 4]
machine:
  concurrent_run: 4,
  cores_per_node: 4,
  name: localhost,
  nodes: 1
runtime:
  program: mpirun
  iterators:
    n_mpi:
      option: '-np '
```

A run and beyond

- More than a single run, PCVS store results under a « database »: **banks**
  - Simplified Git repository
  - Commit = a whole run
  - Branch = run series
- Highlight progression in software developments
  - Easy to use from third-party tools

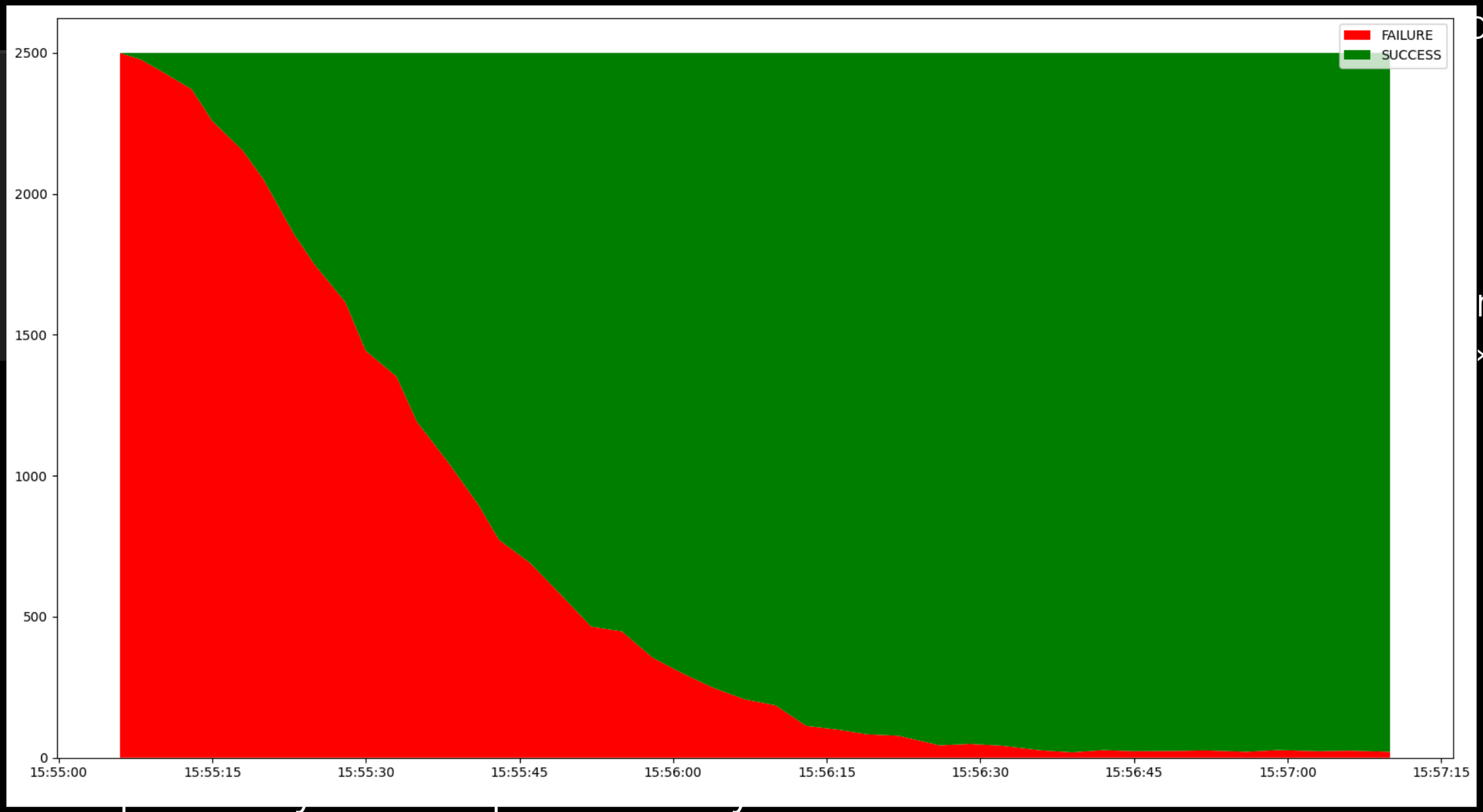
```
└→ pcvs bank show demo

──────────────────────────── BANK VIEW ────────────────────────────
Projects contained in bank '/home/adamj/mnt/work/pcvs/demo.git':
- mpc     : 2 distinct testsuite(s)
  * mpc/d5d3468e3e9a8ec9bba9eb2885434292: 17 run(s)
  * mpc/b6ffe2123be606eab75f75b7dec00eba7d943461: 100 run(s)
- test    : 1 distinct testsuite(s)
  * test/d5d3468e3e9a8ec9bba9eb2885434292: 2 run(s)
- master  : 1 distinct testsuite(s)
  * master: 0 run(s)
- myproject: 1 distinct testsuite(s)
  * myproject/d5d3468e3e9a8ec9bba9eb2885434292: 9 run(s)
```

A run and beyond

```python
from pcvs.dsl import Bank
from pcvs.dsl.analysis import SimpleAnalysis

bank = Bank('demo.git')
a = SimpleAnalysis(bank)

l = bank.list_series()
data = a.generate_weighted_divergence(l[1].name)
data = a.generate_serie_trend(l[1].name)
```

- Per-run test results are relevant but not enough
  - Execution Noise (load, disks..)
  - Binary interpretation
- Analysing test results over multiple runs offer more depth to the « big picture ».
- PCVS provide a DSL to query banks about save runs.
  - Build trends to see the actual progression in development
  - Requalify job status by adding/recompute test results based on measurements.
- Multiple analyses are provided by PCVS
  - Custom can be inserted as plugins

A run and beyond



- Custom can be inserted as plugins

Spack & PCVS

- PCVS is not a Spack package (yet)
- PCVS can interact with Spack in three manners:
  - Install/load packages as deps
  - Validate Spack recipes according to variant matrix
  - Run any test-defined packages (=translating to « spack test »)

```
(pcvs) pcvs run --profile mpi --spack-recipe libfabric

Parallel Computing -- Validation System
Copyright © 2017 -- CEA

INITIALIZATION
❖ Prepare environment
  ◇ Check whether build directory is valid
  ◇ Ensure user-defined programs exist
  ◇ Init & expand criterions
  ◇ Init the global Orchestrator
  ◇ Save Configurations into
❖ Load Test Suites
  ◇ Build test-bases from Spack recipes
❖ ===> Processing done in 0.428 sec(s)
SUMMARY
❖ Global Information
  ◇ Date of execution: Fri Feb  3 11:31:03 2023
  ◇ Run by: Julien Adam <adamj@paratools.com>
  ◇ Active session ID: 231
  ◇ Loaded profile: 'user.mpi'
  ◇ Build stored to:
  ◇ Criterion matrix size per job: 1
  ◇ Bank Management: demo@project
❖ User directories:
  ◇ SPACK: /spack
❖ Globally loaded plugins:
  ◇ TEST_EVAL: MPIPlugin
  ◇ TEST_RESULT_EVAL: BankValidationPlugin
❖ Orchestration infos
  ◇ Test count: 72
  ◇ Max simultaneous Sets: 4
  ◇ Resource count: 1
EXECUTION

Name                    SUCCESS   FAILURE   ERROR   OTHER
spack/libfabric            72        0        0       0
```

S

- PCV
  (yet)
- PCV
  three
  - In
  - Va
    ac
  - Ru
    (=

```
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=gni_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=gni_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=gni_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=gni_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=xpmem_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=xpmem_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=xpmem_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=xpmem_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=efa_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=efa_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=efa_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=efa_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=opx_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=opx_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=opx_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=opx_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=mrail_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=mrail_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=mrail_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=mrail_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=psm2_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=psm2_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=psm2_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=psm2_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=shm_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=shm_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=shm_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=shm_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=sockets_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=sockets_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=sockets_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=sockets_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=cxi_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=cxi_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=cxi_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=cxi_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=tcp_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=tcp_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=tcp_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=tcp_kdreg=True
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=False_fabrics=udp_kdreg=False
SUCCESS ┤ spack/libfabric/libfabric_build_system=autotools_debug=True_fabrics=udp_kdreg=False
```

ROR	OTHER
0	0

Future Work

- Scheduled
  - CLI-based GUI (through `Textualize`)
  - Allow auto-parametrization over compilation tests
  - Job packing into a single allocation
  - Capturing metrics
- Wishlist
  - Job workflow & visualization (conditional run path)
  - Third-party exporters (Prometheus, Graylog)
  - Better Spack CI/test integration
  - Spack & Easybuild support for deployment
  - Users & feedback :)

Thank you for your attention

- ✓ https://pcvs.io/
- ✓ https://pcvs.readthedocs.io/
- ✓ https://github.com/cea-hpc/pcvs
- ✓ julien.adam@paratools.com

ParaTools