Improving Continuous Testing Infrastructure of Xsuite

By: Benchaphorn Chanprasertkul

Supervised By:
Giovanni Iadarola
Szymon Lopaciuk

22 August 2024

TABLE OF CONTENTS

01

Introduction

Xsuite Overview

Python packages

Xsuite is a collection of Python packages designed for simulating beam dynamics in particle accelerators.

Platform

It operates across various computing platforms, including CPUs and GPUs.

The current testing framework

GitHub Actions

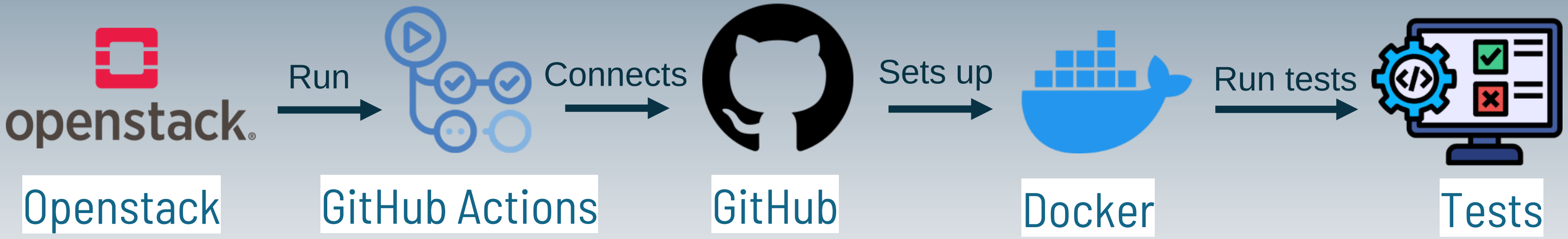
Docker

Openstack

- Utilizes GitHub Actions with Docker or Podman containers to ensure tests run in isolated environments

- CERN-hosted VMs with GPUs on OpenStack, which provide a scalable and flexible testing environment through rapidly configurable virtual machines.

Testing Workflow



02

Background

Testing Challenges

Docker Builds

Docker images were built separately on each test runner, leading to redundant resource use and an opaque process.

Limited Input Capacity in GitHub Dispatch

The GitHub dispatch system was limited to 10 inputs, which restricted our ability to run comprehensive tests. We use these inputs to specify different test packages, and as we added more packages, the 10-input limit became a problem, preventing us from fully testing all component.

Absence of Summary Displays in GitHub Actions

GitHub Actions didn't provide a direct summary of test results, making error identification and debugging more difficult.

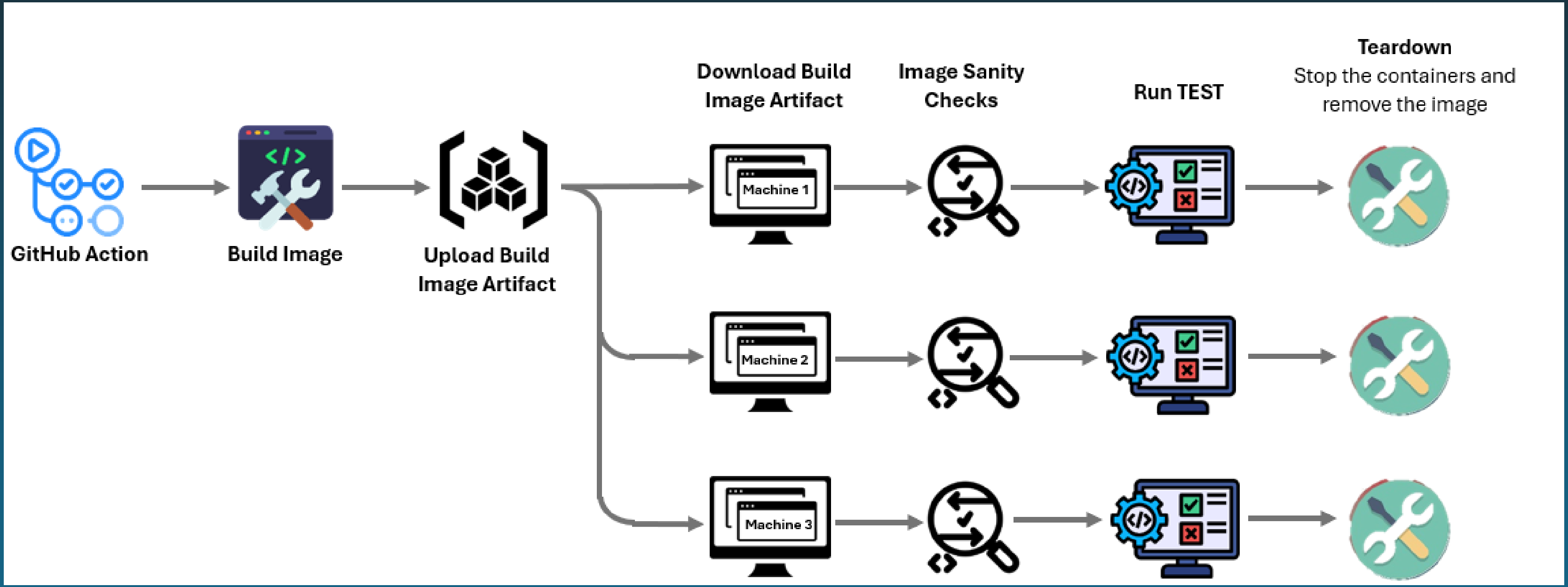
Lack of Consolidated Test Results

There was no automatic summary of test outcomes, making it challenging to quickly assess and track performance metrics over time, we later implemented a solution using HTCondor to run multiple test sessions and gather detailed statistics.

03

Methods

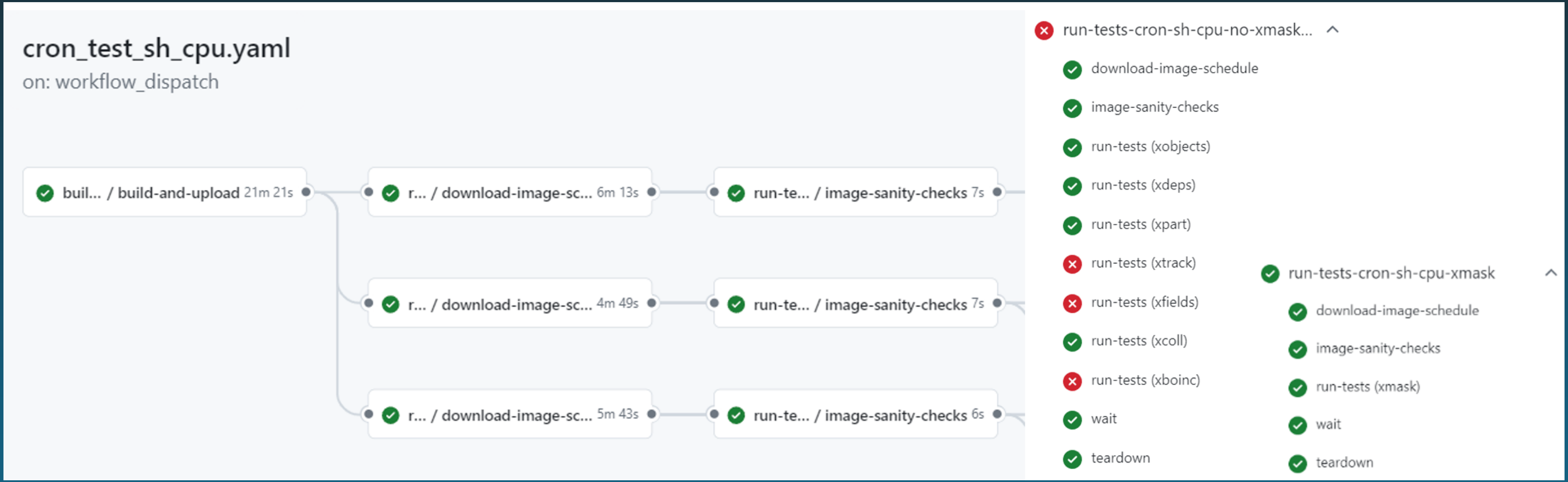
Docker Image Management



GitHub Action
Build Image
Upload Build Image Artifact
Download Build Image Artifact
Image Sanity Checks
Run TEST
Teardown
Stop the containers and remove the image
Machine 1
Machine 2
Machine 3

Fig 1 : Pipeline of Docker Image Management.

Example Use Case (Xsuite Daily testing)


cron_test_sh_cpu.yaml
on: workflow_dispatch

✓ buil... / build-and-upload 21m 21s

✓ r... / download-image-sc... 6m 13s
✓ run-te... / image-sanity-checks 7s

✓ r... / download-image-sc... 4m 49s
✓ run-te... / image-sanity-checks 7s

✓ r... / download-image-sc... 5m 43s
✓ run-te... / image-sanity-checks 6s

✗ run-tests-cron-sh-cpu-no-xmask...
- ✓ download-image-schedule
- ✓ image-sanity-checks
- ✓ run-tests (xobjects)
- ✓ run-tests (xdeps)
- ✓ run-tests (xpart)
- ✗ run-tests (xtrack)
- ✗ run-tests (xfields)
- ✓ run-tests (xcoll)
- ✗ run-tests (xboinc)
- ✓ wait
- ✓ teardown

✓ run-tests-cron-sh-cpu-xmask
- ✓ download-image-schedule
- ✓ image-sanity-checks
- ✓ run-tests (xmask)
- ✓ wait
- ✓ teardown

Fig 2 : Use Case of Xsuite Daily testing.

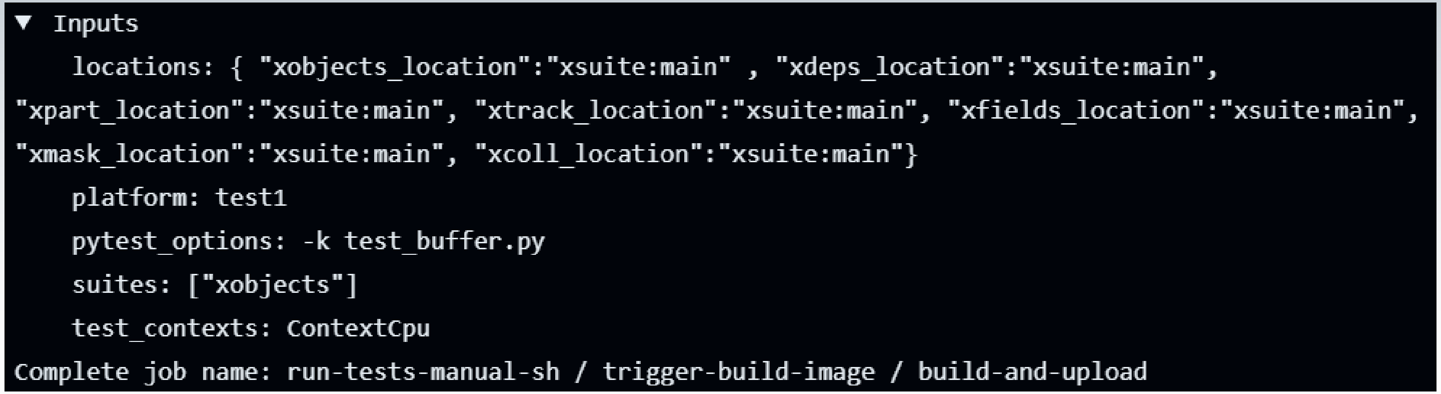
Testing Execution

Flexible Test Configuration

Utilize a JSON configuration approach to address limitations in workflow input capacity, Allowing for easy adjustment of test parameters and the inclusion of various Xsuite packages without the core workflow.

Subset Testing

The manual test execution process was enhanced by enabling the specification of subsets of tests though arbitrary options passed to pytest.



```
▼ Inputs
    locations: { "xobjects_location":"xsuite:main" , "xdeps_location":"xsuite:main",
"xpart_location":"xsuite:main", "xtrack_location":"xsuite:main", "xfields_location":"xsuite:main",
"xmask_location":"xsuite:main", "xcoll_location":"xsuite:main"}
    platform: test1
    pytest_options: -k test_buffer.py
    suites: ["xobjects"]
    test_contexts: ContextCpu
Complete job name: run-tests-manual-sh / trigger-build-image / build-and-upload
```

Fig 3 : Input of subsets testing.

Automated Analytics Tools

Automated Error Detection and Reporting

GitHub Actions now automatically extracts and summarizes errors in **Markdown** format within the workflow. Failures are highlighted and detailed in a collapsible section, improving error visibility and simplifying debugging.

Test Result Analysis Script

A custom script analyzes test outcomes, generating metrics like pass/fail rates and test durations in CSV files. It tracks non-deterministic test failures and identifies slow tests, aiding in performance optimization and proactive issue resolution.

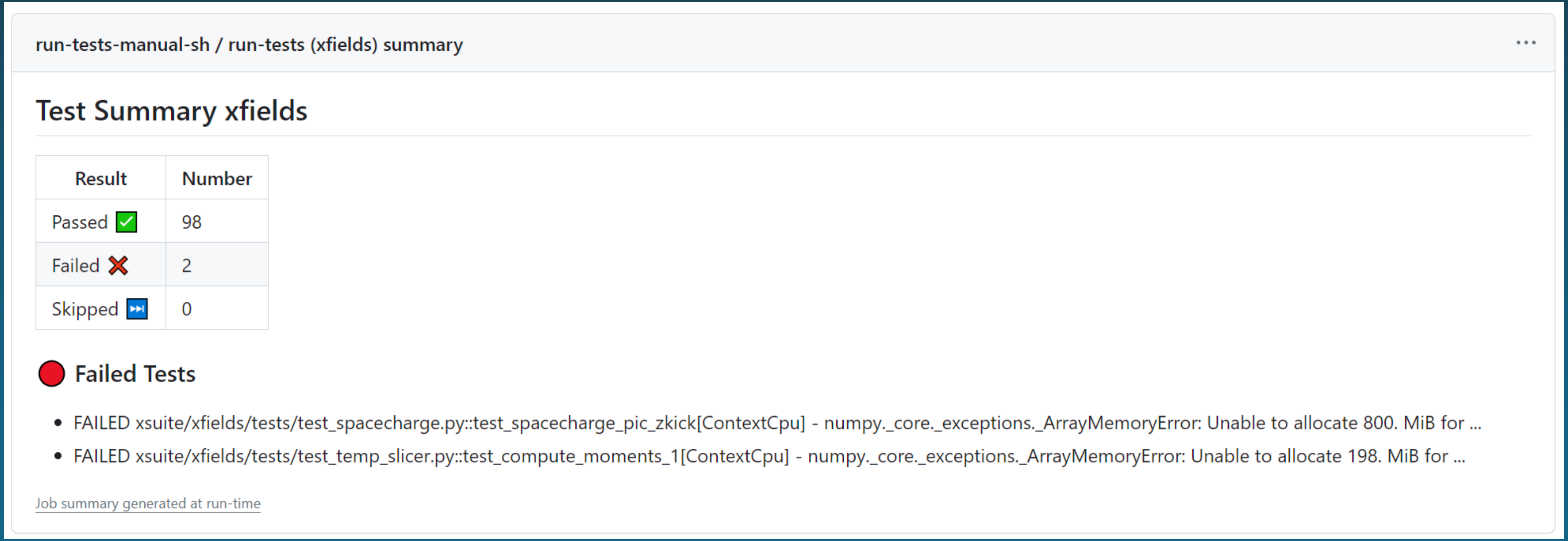
04

Results and Conclusion

Results and Conclusion

Automated Error Detection and Reporting

GitHub Actions now automatically extracts and summarizes errors in Markdown format within the workflow. Failures are highlighted and detailed in a collapsible section, improving error visibility and simplifying debugging.



run-tests-manual-sh / run-tests (xfields) summary

Test Summary xfields

Result	Number
Passed ✅	98
Failed ❌	2
Skipped ⏭️	0

🔴 Failed Tests

- FAILED xsuite/xfields/tests/test_spacecharge.py::test_spacecharge_pic_zkick[ContextCpu] - numpy._core._exceptions._ArrayMemoryError: Unable to allocate 800. MiB for ...
- FAILED xsuite/xfields/tests/test_temp_slicer.py::test_compute_moments_1[ContextCpu] - numpy._core._exceptions._ArrayMemoryError: Unable to allocate 198. MiB for ...

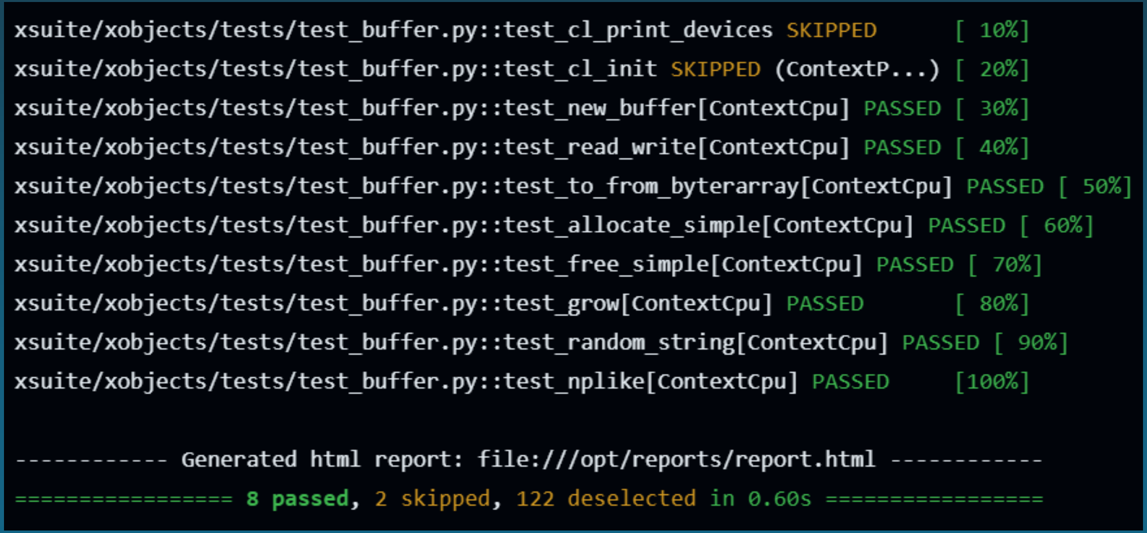
Job summary generated at run-time

Fig 4 : Summarizes errors in Markdown format .

Results and Conclusion

Flexibility Improvements

JSON-based configurations and subset testing allowing for more efficient and targeted testing through GitHub Actions.



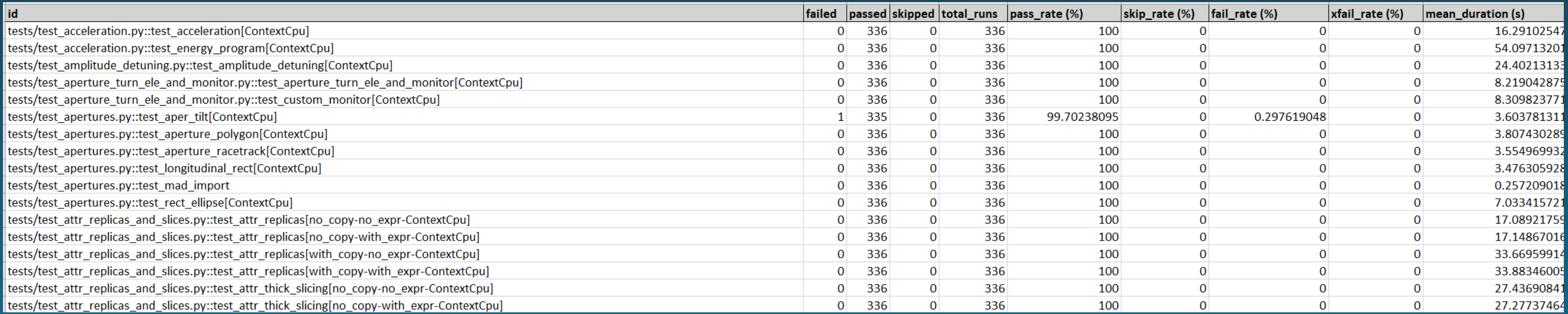
```
xsuite/xobjects/tests/test_buffer.py::test_cl_print_devices SKIPPED      [ 10%]
xsuite/xobjects/tests/test_buffer.py::test_cl_init SKIPPED (ContextP...) [ 20%]
xsuite/xobjects/tests/test_buffer.py::test_new_buffer[ContextCpu] PASSED [ 30%]
xsuite/xobjects/tests/test_buffer.py::test_read_write[ContextCpu] PASSED [ 40%]
xsuite/xobjects/tests/test_buffer.py::test_to_from_byterarray[ContextCpu] PASSED [ 50%]
xsuite/xobjects/tests/test_buffer.py::test_allocate_simple[ContextCpu] PASSED [ 60%]
xsuite/xobjects/tests/test_buffer.py::test_free_simple[ContextCpu] PASSED [ 70%]
xsuite/xobjects/tests/test_buffer.py::test_grow[ContextCpu] PASSED       [ 80%]
xsuite/xobjects/tests/test_buffer.py::test_random_string[ContextCpu] PASSED [ 90%]
xsuite/xobjects/tests/test_buffer.py::test_nplike[ContextCpu] PASSED     [100%]

----------- Generated html report: file:///opt/reports/report.html ------------
================= 8 passed, 2 skipped, 122 deselected in 0.60s =================
```

Fig 5 : Subsets test running.

Results and Conclusion

Analytical Insights

The custom script for test result analysis has proven invaluable. By providing detailed metrics such as pass/fail rates and test durations in an easily accessible CSV and graphical format



id	failed	passed	skipped	total_runs	pass_rate (%)	skip_rate (%)	fail_rate (%)	xfail_rate (%)	mean_duration (s)
tests/test_acceleration.py::test_acceleration[ContextCpu]	0	336	0	336	100	0	0	0	16.29102547
tests/test_acceleration.py::test_energy_program[ContextCpu]	0	336	0	336	100	0	0	0	54.09713201
tests/test_amplitude_detuning.py::test_amplitude_detuning[ContextCpu]	0	336	0	336	100	0	0	0	24.40213133
tests/test_aperture_turn_ele_and_monitor.py::test_aperture_turn_ele_and_monitor[ContextCpu]	0	336	0	336	100	0	0	0	8.219042875
tests/test_aperture_turn_ele_and_monitor.py::test_custom_monitor[ContextCpu]	0	336	0	336	100	0	0	0	8.309823771
tests/test_apertures.py::test_aper_tilt[ContextCpu]	1	335	0	336	99.70238095	0	0.297619048	0	3.603781311
tests/test_apertures.py::test_aperture_polygon[ContextCpu]	0	336	0	336	100	0	0	0	3.807430289
tests/test_apertures.py::test_aperture_racetrack[ContextCpu]	0	336	0	336	100	0	0	0	3.554969932
tests/test_apertures.py::test_longitudinal_rect[ContextCpu]	0	336	0	336	100	0	0	0	3.476305928
tests/test_apertures.py::test_mad_import	0	336	0	336	100	0	0	0	0.257209018
tests/test_apertures.py::test_rect_ellipse[ContextCpu]	0	336	0	336	100	0	0	0	7.033415721
tests/test_attr_replicas_and_slices.py::test_attr_replicas[no_copy-no_expr-ContextCpu]	0	336	0	336	100	0	0	0	17.08921759
tests/test_attr_replicas_and_slices.py::test_attr_replicas[no_copy-with_expr-ContextCpu]	0	336	0	336	100	0	0	0	17.14867016
tests/test_attr_replicas_and_slices.py::test_attr_replicas[with_copy-no_expr-ContextCpu]	0	336	0	336	100	0	0	0	33.66959914
tests/test_attr_replicas_and_slices.py::test_attr_replicas[with_copy-with_expr-ContextCpu]	0	336	0	336	100	0	0	0	33.88346005
tests/test_attr_replicas_and_slices.py::test_attr_thick_slicing[no_copy-no_expr-ContextCpu]	0	336	0	336	100	0	0	0	27.43690841
tests/test_attr_replicas_and_slices.py::test_attr_thick_slicing[no_copy-with_expr-ContextCpu]	0	336	0	336	100	0	0	0	27.27737464

Fig 6 :Results of tests in CSV files.

Results and Conclusion

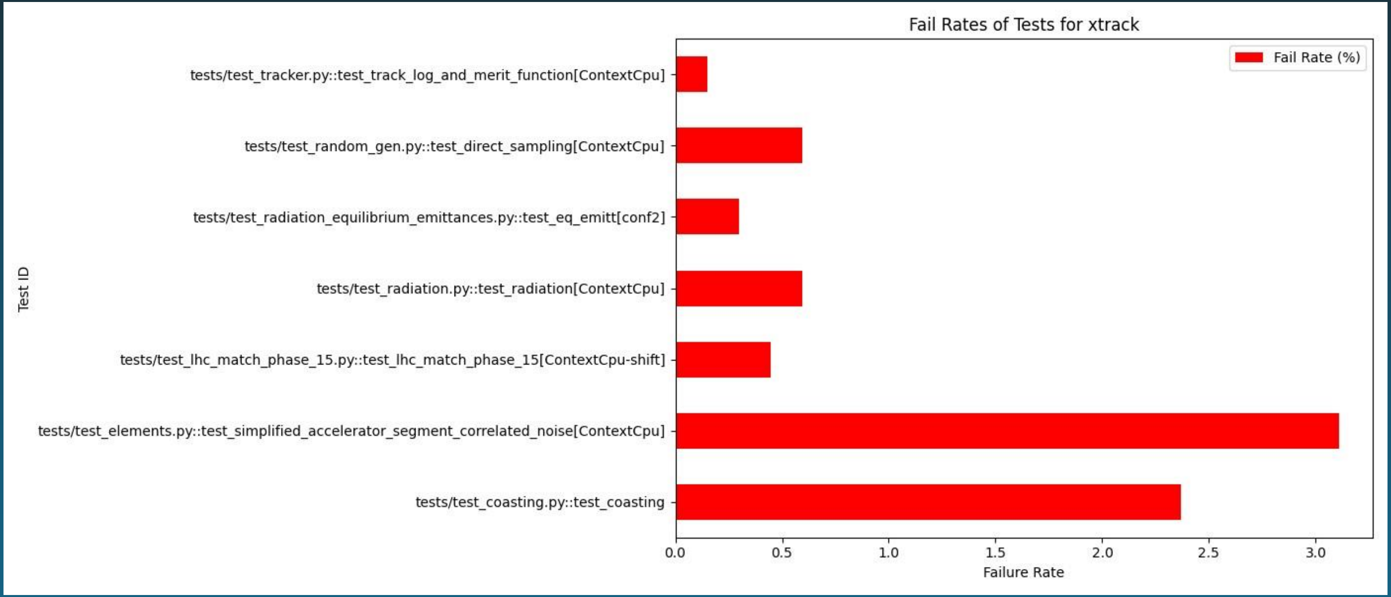


Fig 7 :Fail rates of xtrack with 675 sample size.

Results and Conclusion

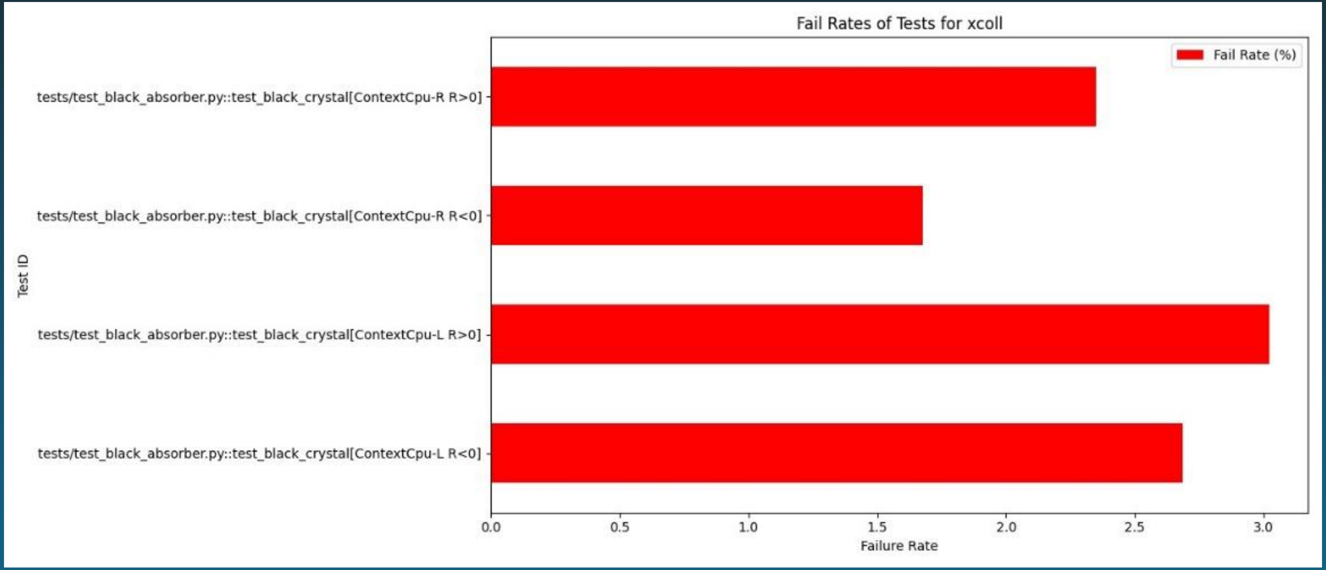


Fig 8 : Fail rates of xcoll with 298 sample size.

Results and Conclusion

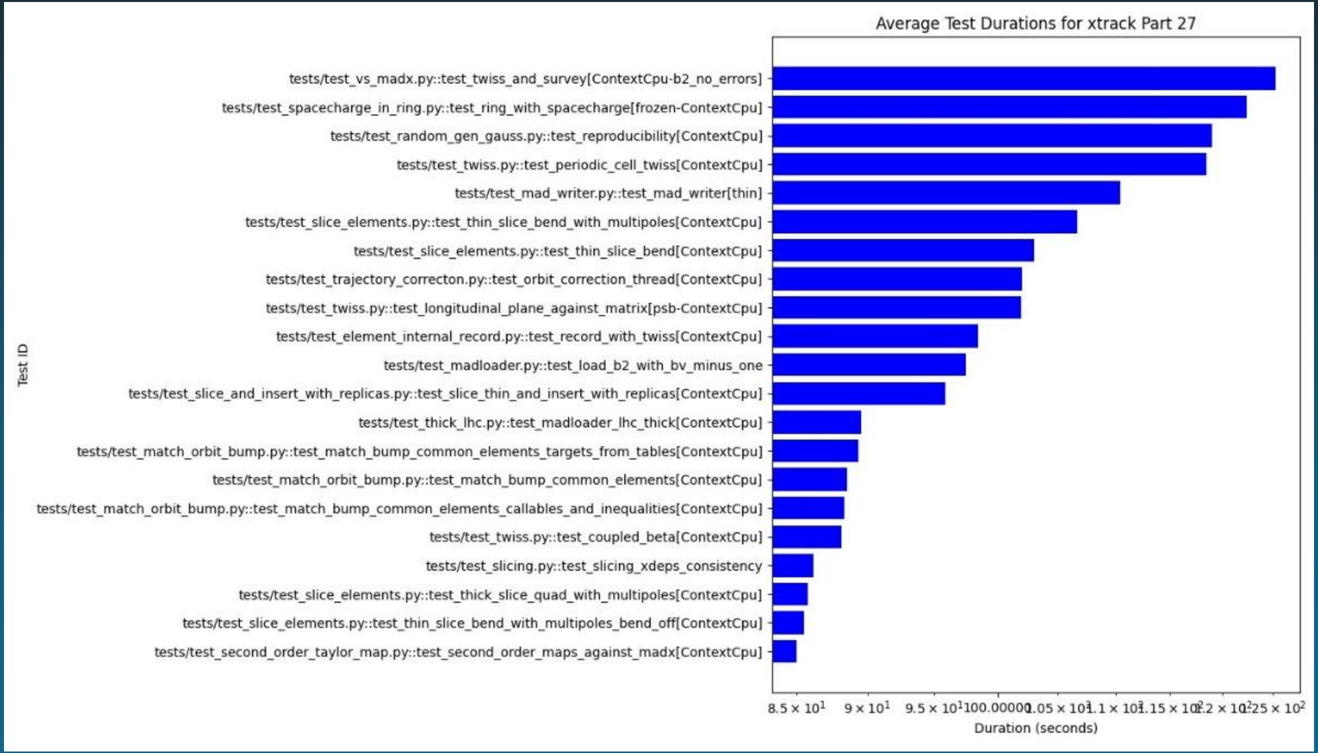


Fig 9 : Sorted Average Test Durations for 20 Tests with Durations in the xtrack Package.

Results and Conclusion

Limitations

Storage Limitations : GitHub’s 90-day limit on logs and 500 artifact cap can restrict long-term data access, but custom retention policies help manage this.

Future Work

Using CERN tools to build the image: Integration of CERN's GitLab CI/CD infrastructure into our processes.

Thank You for your Attention