



Validation Status and Roadmap

Dmitry Kalinkin

University of Kentucky

Key components

- » Development tests for epic geometry and for EICrecon
- » Detector benchmarks
- » Physics benchmarks

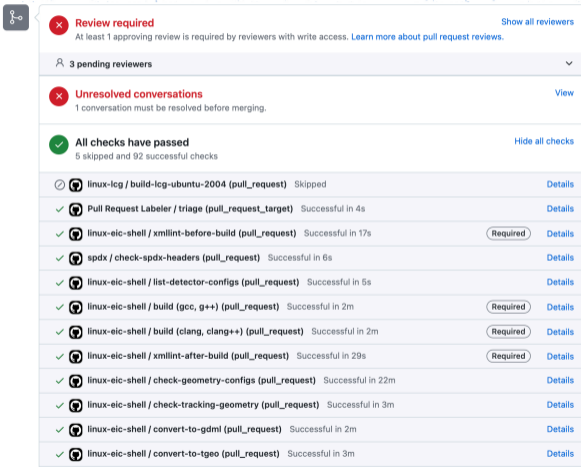
required contribution,
strict standards,
early feedback
low statistics



voluntary contribution,
relaxed standards,
late feedback,
large statistics

epic geometry tests

https://github.com/eic/epic/pulls



The screenshot shows a GitHub pull request interface. At the top, there is a red 'Review required' status with a sub-message: 'At least 1 approving review is required by reviewers with write access. Learn more about pull request reviews.' Below this, it indicates '3 pending reviewers'. A red 'Unresolved conversations' status follows, stating '1 conversation must be resolved before merging.' A green 'All checks have passed' status is shown next, with '5 skipped and 92 successful checks'. A list of checks follows, including 'linux-icg / build-icg-ubuntu-2004 (pull_request)' which is skipped, and several 'linux-eic-shell' checks for linting, headers, detector configs, building with gcc/g++, clang/clang++, and geometry configurations, all of which are successful. Some checks are marked as 'Required'.

Check Name	Status	Duration	Details
linux-icg / build-icg-ubuntu-2004 (pull_request)	Skipped		Details
Pull Request Labeler / triage (pull_request_target)	Successful	4s	Details
linux-eic-shell / xmllint-before-build (pull_request)	Successful	17s	Required Details
spdx / check-spdx-headers (pull_request)	Successful	6s	Details
linux-eic-shell / list-detector-configs (pull_request)	Successful	5s	Details
linux-eic-shell / build (gcc, g++) (pull_request)	Successful	2m	Required Details
linux-eic-shell / build (clang, clang++) (pull_request)	Successful	2m	Required Details
linux-eic-shell / xmllint-after-build (pull_request)	Successful	29s	Required Details
linux-eic-shell / check-geometry-configs (pull_request)	Successful	22m	Details
linux-eic-shell / check-tracking-geometry (pull_request)	Successful	3m	Details
linux-eic-shell / convert-to-gdml (pull_request)	Successful	2m	Details
linux-eic-shell / convert-to-tgeo (pull_request)	Successful	3m	Details

- » Compile with gcc and clang
- » Run TGeo and Geant4 overlap checks for all configurations
- » Run ACTS checks
- » Produce GDML, ROOT(TGeo) geometry files
- » Render dawn views
- » Trigger running of detector and physics benchmarks on eicweb, status is reported back

epic geometry tests

<https://github.com/eic/epic/pulls>

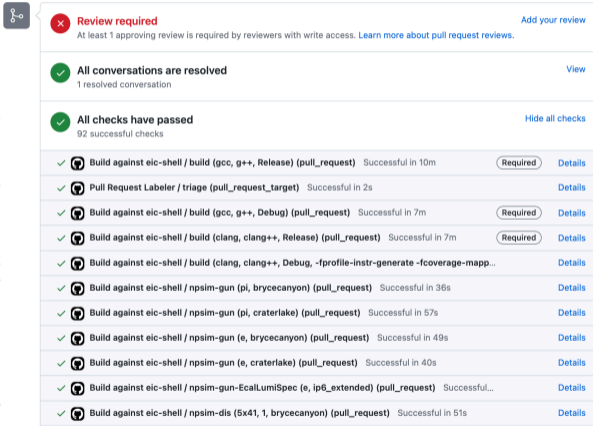
The screenshot shows a GitHub Actions workflow named 'add diffractive_vmp' that has successfully completed. The pipeline was triggered by Wouter Deconinck for commit fd283781, finished 2 weeks ago. It consists of 71 jobs, with a total duration of 45 minutes and 12 seconds. The workflow is divided into four stages: simulate, reconstruct, analyze, and collect. Each stage contains several jobs, all of which are marked as successful with green checkmarks.

simulate	reconstruct	analyze	collect
backgrounds:synchrotron:simulate	single:reconstruct	single:analyze	backgrounds:resu
diffractive_vm:simulate			diffractive_vm:res
dis:simulate			dis:results
dvcs:simulate			dvcs:results
single:simulate			single:results
tcs:simulate			tcs:results
u_omega:simulate			u_omega:results

- » Compile with gcc and clang
- » Run TGeo and Geant4 overlap checks for all configurations
- » Run ACTS checks
- » Produce GDML, ROOT(TGeo) geometry files
- » Render dawn views
- » Trigger running of detector and physics benchmarks on eicweb, status is reported back

EICrecon tests

<https://github.com/eic/EICrecon/pulls>



Review required [Add your review](#)
At least 1 approving review is required by reviewers with write access. [Learn more about pull request reviews.](#)

All conversations are resolved [View](#)
1 resolved conversation

All checks have passed [Hide all checks](#)
92 successful checks

✔	Build against eic-shell / build (gcc, g++, Release) (pull_request)	Successful in 10m	Required	Details
✔	Pull Request Labeler / triage (pull_request_target)	Successful in 2s		Details
✔	Build against eic-shell / build (gcc, g++, Debug) (pull_request)	Successful in 7m	Required	Details
✔	Build against eic-shell / build (clang, clang++, Release) (pull_request)	Successful in 7m	Required	Details
✔	Build against eic-shell / build (clang, clang++, Debug, -fprofile-instr-generate -fcoverage-mapp...			Details
✔	Build against eic-shell / npsim-gun (pi, brycecanon) (pull_request)	Successful in 36s		Details
✔	Build against eic-shell / npsim-gun (pi, craterlake) (pull_request)	Successful in 57s		Details
✔	Build against eic-shell / npsim-gun (e, brycecanon) (pull_request)	Successful in 49s		Details
✔	Build against eic-shell / npsim-gun (e, craterlake) (pull_request)	Successful in 40s		Details
✔	Build against eic-shell / npsim-gun-EcalLumiSpec (e, ip6_extended) (pull_request)	Successful...		Details
✔	Build against eic-shell / npsim-dis (5x41, 1, brycecanon) (pull_request)	Successful in 51s		Details

- » Compile with gcc and clang
- » Static analysis and code style (clang-tidy, IWYU)
- » With AddressSanitizer and UBSanitizer
- » Run unit tests
- » Run simulation and reconstruction for gun and DIS (100 events)
- » Run JANA-based benchmarks
- » Upload artifacts (EDM4hep sim, EDM4eic reco, jana factory parameters, janadot, coverage report, doxygen)
- » Compare to reco EDM4eic to artifact from the base branch

epic-capybara

Experimental set of tools to:

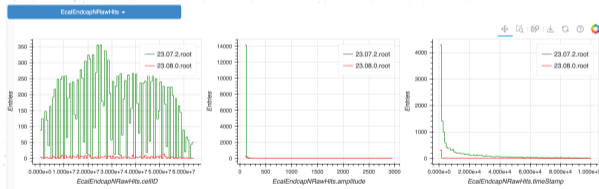
- » Download GitHub artifacts
- » Compare PODIO files
- » Visualize comparisons
(Each branch histogrammed)
- » Share (upload to GitHub pages)

Rather not have to
develop/maintain those ourselves...

<https://github.com/eic/epic-capybara>

Demo: regression in 23.08 campaign due to
incorrect introduction of thresholds

<https://veprbl.github.io/capybara-reports/>



ELCrecon-specific tasks

- Full unit test coverage for algorithms
 - Implement mock geometry services
 - Test cases for tracking
 - Test cases for calorimetry
 - Test cases for PID
 - Test cases for event-level reconstruction
- Check for some invariants (e.g units rescaling, compiler switching)

Common benchmarking tasks

- » Benchmark codes
 - Adopt a workflow execution system
(Local execution, Caching, Input requirements spec)
 - Adopt a (non-graphic) output format for histograms/profiles
 - Support re-running with non-standard software dependency versions
(What is the effect of bumping DD4hep version?)
 - Aggregation of historical data
(Software and key detector performance metrics)
- Tool for presentation/comparisons of benchmark results (like “rivet-mkhtml” and like “mlflow”)
(See also talk by Torri)
 - UI to explore and select available samples (time series, commit series, dependency version series)
 - UI for presentation of comparisons between two samples
 - UI for presentation of summaries along many samples

Tasks for detector benchmarks

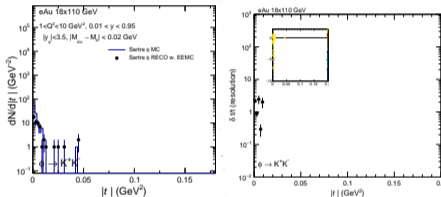
» Items from the “common” list

- Implement a reference benchmark code
- Generate artifacts to keep reconstruction up to date:
 - Up to date calibrations for calorimeters
 - Tracking material map
 - Reproducible ML artifacts
(e.g. far-backward and far-forward tracking, AstroPix PID)
- Implement simple pass/fail conditions for geometry development

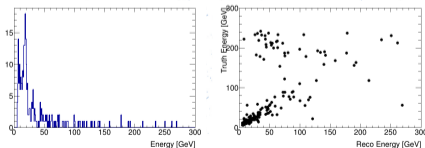
New physics benchmarks

After call to PWGs we've received two new cool benchmarks. Now running on eicweb.

Diffraction vector meson production in pAu



Jet benchmarks (here for DIS 18×275 , $Q^2 > 100 \text{ GeV}^2$)



Tasks for physics benchmarks

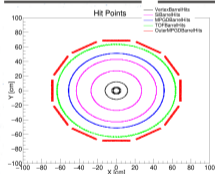
» Items from the “common” list

- Implement map/reduce steps to enable big data (campaign processing)
- Implement a reference benchmark code
 - Things make take a turn towards accepting benchmarks of semi-arbitrary format. Almost like providing Reana-like service.
- Submit eicweb pipelines from the EICrecon repo
- Automation for running over the campaign

Volunteered benchmarks

People like to contribute their code where there is foot traffic

- » fhcal_studiesProcessor in EICrecon \approx 1000 LoC
Ran within the reconstruction, if we can't access all necessary structures to analyze the data, it's an issue with our EDM
- » Plot_eta.C, draw_Performance.C, draw_hits.C in EICrecon



- » https://github.com/eic/epic/blob/main/scripts/subdetector_tests/material_scan.py

Desired properties

- » Practical applications
 - ~~inflicting good upon the world~~
 - aid in software change review process
 - for data analyzers by data analyzers
- » Campaign readiness at all times
 - reduce monthly crunch time
 - avoid buggy productions
- » Shared infrastructure
 - embrace contributions from the collaboration at large
 - common resources: GitHub runners, OSG
- » Minimal development for tooling
 - doing things in-house has shortcomings
 - pressing schedule to deliver
 - existing solutions to reuse?

Conclusions

- » Big progress with fast CI validation for epic and EICrecon
- » Some decisions still need to be made to get benchmarks “ready for production”
- » Infrastructure needs to mature fast
...for us to effectively direct the emerging efforts towards the consolidated benchmarking

Backup

ePIC benchmarks on eicweb

ePIC had inherited benchmarks from Athena:

- » https://eicweb.phy.anl.gov/EIC/benchmarks/detector_benchmarks
- » https://eicweb.phy.anl.gov/EIC/benchmarks/physics_benchmarks
- » https://eicweb.phy.anl.gov/EIC/benchmarks/reconstruction_benchmarks

Something to learn from!

- » Running on the grid after each software change (Continuous Integration)
- » Transparent procedures – source code available
- » Unfortunately, analysis and interface are unsophisticated
- » Not friendly to deadline-driven development - no user adoption

```

b0_tracker
+-- analysis
|   +-- b0_tracker_hits.cxx
+-- scripts
    +-- gen_forward_protons.cxx
barrel_ecal/scripts
    +-- emcal_barrel_energy_scan_analysis.cxx
    +-- emcal_barrel_particles_analysis.cxx
    +-- emcal_barrel_pi0_analysis.cxx
    +-- emcal_barrel_pion_rejection_analysis.cxx
    +-- emcal_barrel_pions_analysis.cxx
barrel_hcal/scripts
    +-- hcal_barrel_energy_scan_analysis.cxx
    +-- hcal_barrel_particles_analysis.cxx
material_maps
+-- scripts
others
+-- materialScanEta.cxx
+-- materialScanEtaPhi.cxx
pid/scripts
    +-- drich_analysis.cxx           // INCOMPLETE
    +-- mrich_analysis.cxx         // INCOMPLETE
timing
tracking_detectors
+-- analysis
|   +-- sim_track_hits.cxx
+-- scripts
    +-- test_matscan.cxx
    +-- matscan_plot.py
zdc
+-- scripts
|   +-- analysis_zdc_particles.cxx
+-- simple_checking.cxx
+-- simple_info_plot_histograms.cxx
+-- zdc_neutrons_reader.cxx

```

```

backgrounds
+-- analysis
    +-- synchrotron_raw.cxx // EMPTY
    +-- synchrotron_sim.cxx // NO OUTPUT
diffractive_vm
+-- analysis
    +-- diffractive_vm.cxx // NEW
dis
+-- analysis
|   +-- dis_electrons.cxx
    +-- jets.cxx           // NEW
    +-- kinematics_correlations.py
    +-- truth_reconstruction.py
dvcs
+-- analysis
    +-- dvcs_ps_gen.cxx
    +-- dvcs_tests.cxx
dvmp
+-- analysis
|   +-- vm_invar.cxx
|   +-- vm_mass.cxx
single
+-- analysis
    +-- analyze.cxx
tcs
+-- analysis
    +-- tcs_tests.cxx
u_omega
+-- analysis
    +-- demo.cxx           // INCOMPLETE

```