Validation in ePIC: status and prospects of tooling

Dmitry Kalinkin

July 12, 2023

Past: ATHENA benchmarks

The "eicweb" GitLab CI instance hosted at ANL is setup to run CI checks for development versions of the ePIC geometry repository `https://github.com/eic/epic`, but not for EICrecon.

- Detector benchmarks
  `https://eicweb.phy.anl.gov/EIC/benchmarks/detector_benchmarks`
  Per-detector studies on ddsim simulation output
- Reconstruction benchmarks `https://eicweb.phy.anl.gov/EIC/benchmarks/reconstruction_benchmarks`
  Mainly runs different subsets of reconstruction algorithms from Juggler
- Physics benchmarks
  `https://eicweb.phy.anl.gov/EIC/benchmarks/physics_benchmarks`
  Runs few studies that look at DIS, DVCS, DVMP, TCS physics processes and at single tracks for reconstructed output from EICrecon (only the released version). Examples with both C++ (RDataFrame) and Python (uproot).

Artifacts uploaded to eicweb GitLab.

Continuous Integration for EICrecon

GitHub Actions on public runners

- Does builds with GCC and Clang
- Provides static analysis (clang-tidy)
- Runtime analysis (currently AddressSanitizer and LeakSanitizer, work needed to fix for UBSanitizer)
- Framework for unit tests with code coverage
  Need to contribute more unit tests. Providing mock objects DD4hep and Acts geometries is a major roadblock.
- Runs EICrecon on a few single e^-, e^- pair in EcalLumiSpec and DIS simulated samples (100 events each)
- Output files are not currently analyzed
- Many useful artifacts (*.edm4eic.root, *.json) are uploaded to GitHub

Why validation

- It's not that we don't know how to debug our software
- **Catching errors early and at predictable times saves resources that we could spend on doing interesting research**
- "Validation" can not be provided externally (as a service), collaboration as a whole would need to learn some practices
- Centralized validation allows to create and share knowledge about invariants and guarantees about the software

Continuous Intergration: Needs

- A lot of regressions can be catched by diffing PODIO files
  I have some snippets of code, but extra practical experience and effort is needed to come up with a proper tool
- Tooling to compare benchmark results
  - A dashboard to browse and compare recorded metrics (like MLflow but for CI)
  - No known ready solutions, nice data presentation frameworks available Plotly/Dash, Bokeh, Vega(Lite)
  - Preference towards a public infrastructure (GitHub Actions, GitHub Pages) to open up development. Interesting examples:
    `https://acts-project.github.io/metrics/metric/compile_max_rss/max_rss_Examples_Algorithms_TrackFittingChi2_src_TrackFittingChi2AlgorithmFunction.cpp/`
    `https://github.com/benchmark-action/github-action-benchmark`
- Implementation of detector benchmarks for ePIC geometry
- Implementation of reconstruction benchmarks for EICrecon

Defensive programming in EICrecon: Wishlist

Some validation needs to be applied to non-scripted scenarios of end-user work

- Don't use exceptions for control flow ⇒ Don't ignore exceptions
- Don't pollute output logs with noise
- Parameter validation (validate values, check if non-existent parameter names are set)
- Don't silently disable factories on missing inputs (detect when invalid collection names are set)
- Check consistency between simulation and reconstruction geometry version (geometry checksumming, efforts by Wouter et al.)

Physics benchmarks

Continuous Delivery for Yellow-Report-style studies. Few challeges:

- Large volumes of data (e.g. a 5 Tb DIS sample)
- Large number of analyses
- Many potential analyzers potentially seeking to contribute, but have to start from scratch
- `https://github.com/eic/epic-analysis` perhaps the most mature framework (personally, haven't looked into it)
- Physics benchmarking effort can lay foundation for "best" analysis practices
- Reusing CI tooling means extra requirements on compatibility with different infrastructure (OSG/SDCC/ifarm)

Things impossible with the current technology:

- Unit-safety (3+ unit systems: DD4hep/TGeo, EDM, Acts)
- Non-centralized A/B testing with non-standard/patched versions of dependencies
- ...?