Automated workflow of end-to-end simulations and optimizing detector designs

Whitney Armstrong, Manoj Jadhav, Sylvester Joosten, Jihee Kim, Zein-Eddine Meziani, Chao Peng, Tomas Polakovic, Marshall Scott, Junqi Xie
Argonne National Laboratory

Zhenyu Ye, Ziyue Zhang (UIC)

March 19, 2021

Argonne NATIONAL LABORATORY

Overview

- Software R&D Overview
- Software Toolkit
  - Input data
  - Geometry Definition and Geant4 Simulation
  - Digitization and Reconstruction
  - Containerization
- `eicweb.phy.anl.gov` CI/CD Pipelines
- New paradigm in simulation workflow
- Future Directions and Challenges
- Summary

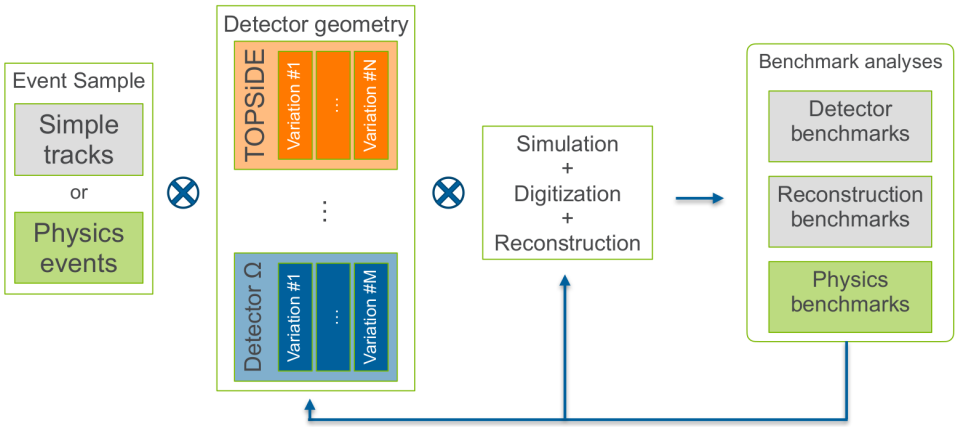
Software tools selected:

- Externally supported and maintained
- Integrate easily with other components
- Take a modern approach
- Leveraging modern, actively supported tools frees up time to focus on forward-looking workflows and interfaces

Not the focus of this talk

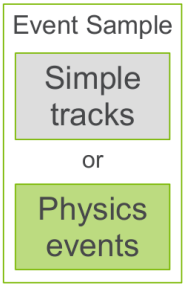
This work is supported by Argonne's LDRD program.

Physics driven detector benchmark and optimization



Event Generation and Data Inputs

Standardized around HepMC3 data format

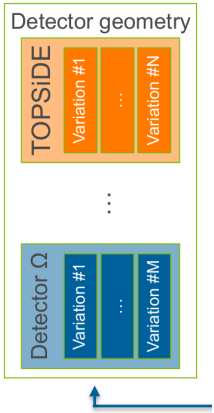


Twofold purpose:

- Validate detector and benchmark reconstruction chain (using particle guns)
- Characterize and benchmark detector setup for desired physics observables (using custom event generators). Towards a fully integrated setup for on-demand event generation to cover all NAS requirements for EIC Software chain can work with any event sample in HepMC3 format

Detector Geometry

Parametrized detector implementations with DD4hep



- Parametrized detector description defined using DD4hep
- NPDet: an extension for DD4hep that adds parametrized detectors for NP experiments (e.g. Cherenkov counters) and various dd4hep-based tools (visualization, conversion, inspection, etc.)
- **Single, unique source for geometry information**: dispatches full GEANT4 simulation and provides geometry for reconstruction
- Aim towards library of configurable detector options to feed into benchmark & optimization process.

Simulation, Digitization and Reconstruction

Core toolkit

- DD4hep: Geant4 geometry fully defined through detector plugin libraries, provides wrappers and tools to run geant4
- ACTS: Experiment-independent tracking toolkit developed for HEP in modern C++, ACTS' geometry constructed from DD4hep via plugin
- GAUDI: Generic open project for building event processing frameworks, developed at CERN, used by e.g. LHCb & ATLAS Enables modern task-based concurrent execution in a heterogeneous computing environment
- Project Juggler: Prototype event processing framework for EIC, combining GAUDI, ACTS, DD4hep and Podio. Approach allows us to focus on the hard part (writing algorithms), while leveraging modern tools

Data model for EIC: eicd

Plain old data model definition with podio

```
      - long long amplitude // The amplitude of the hit in ADC counts.
      - long long timeStamp // The time stamp for the hit.

  eic::CalorimeterHit:
    Description: "Calorimeter hit"
    Author : "W.Armstrong"
    Members:
      - long long           cellID    // The detector specific (geometrical) cell id.
      - float               energy    // The energy of the hit in [GeV].
      - float               time      // The time of the hit in [ns].
      - eic::VectorXYZ      position  // The global position of the hit in world coordinates.
      - eic::VectorXYZLocal local     // The local position of the hit in detector coordinates.
      - eic::DimensionXYZ   dimension // The dimension information of the cell
      - int                 type      // The type of the hit.
    OneToOneRelations:
      - eic::RawCalorimeterHit rawHit // The RawCalorimeterHit
```

```
namespace eic {

class CalorimeterHitData {
public:
  long long cellID;
  float energy;
  float time;
  ::eic::VectorXYZ position;
  ::eic::VectorXYZLocal local;
  ::eic::DimensionXYZ dimension;
  int type;
};

} // namespace eic
```

```
namespace eic {

class CalorimeterHit;
class ConstCalorimeterHit;

class CalorimeterHitObj : public podio::ObjBase {
public:
  CalorimeterHitObj();
  CalorimeterHitObj(const CalorimeterHitObj&);
  CalorimeterHitObj(const podio::ObjectID id, CalorimeterHitData data);
  virtual ~CalorimeterHitObj();

public:
  CalorimeterHitData data;
  ::eic::ConstRawCalorimeterHit* m_rawHit;
};

} // namespace eic
```

- Robust flexible plain-old-data types using Podio.
- Data model defined in a simple Yaml file.
- Models (`dd4pod` and `eicd`) isolate running of geant4 and Juggler
- New additions and modifications made easy and complication free with modern workflow (more on this later).

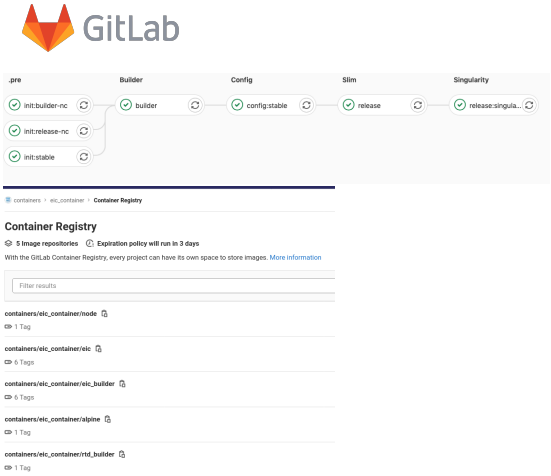
Observation: Podio's usefulness comes from simple yaml specification for defining data model

New tools can be quickly developed using the parsed yaml file as the blueprint: allows new languages, memory layouts, etc.

Building the software

Containerization

- Leverage own GitLab server instance, with continuous integration (CI) to automatically build container
- Build Docker container for use in CI toolchain
- Build Singularity container for use in HPC environments (and local development)



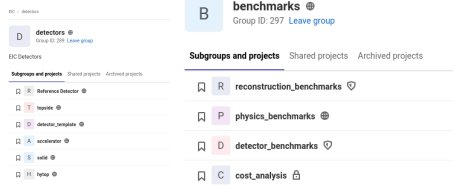
New paradigm in simulation workflow

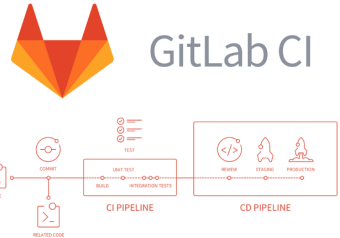
Pushing CI/CD into new territory

First some jargon

- CI/CD: Continuous Integration/Continuous Deployment or Delivery
- pipeline: a collection of jobs run in a certain order
- job: A script that runs in a CI pipeline
- artifact: file(s) produced by a job(s) uploaded to gitlab (at most 10s of MB)
- pipeline trigger: connecting pipelines

- Making use of gitlab groups to organize repositories.

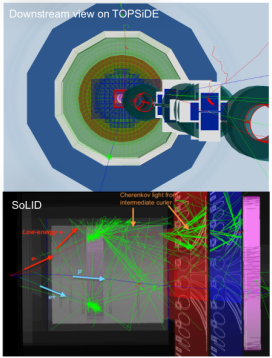




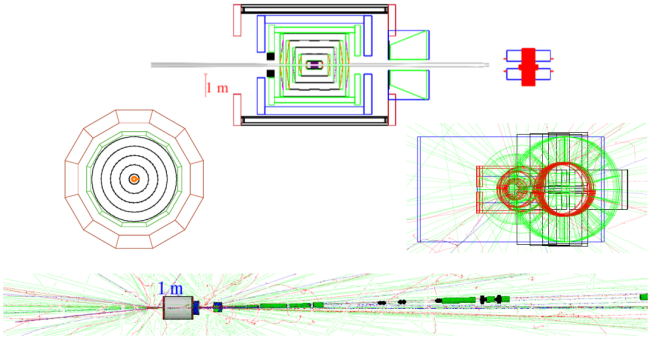
Detector Pipelines

- Automatic visualizations →save as job artifacts
- Check for geometry overlaps →runs on every merge request

Web-based event display based on ROOT/TGeo



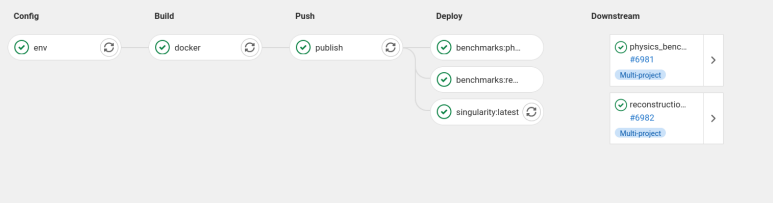
"Engineering" detector cutouts and event display with Dawn/Dawncut



New paradigm in simulation workflow

Connecting Pipelines

A merge request to juggler triggers its pipeline:



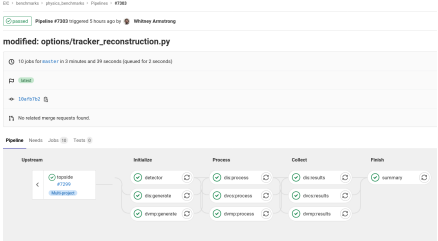
which then causes the reconstruction and physics benchmarks to run any job failures propagate back to original pipeline.

Updating an algorithm

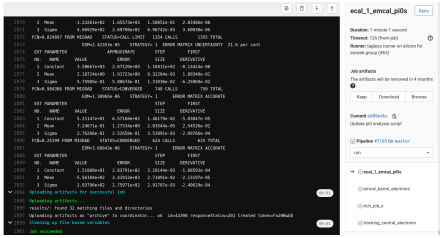
Because full chain is triggered a merge request, running its own pipeline(s), breaking changes at any stage will be noticed.

More Pipelines

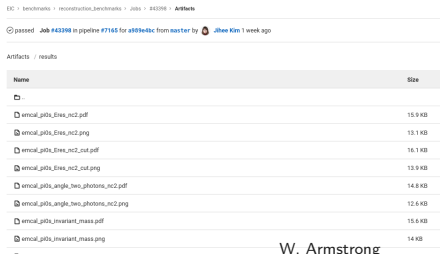
Pipeline from upstream detector trigger



Real-time job console output



Browsable artifacts (also accessible via API)



- With UIC collaborators, we have tested/debugged the workflow.
- Learned a lot of what works and what doesn't work
- Example: why work locally when full chain can be triggered via merge request pipelines?
- Develop and run end-to-end full simulations from your web browser!

Argonne NATIONAL LABORATORY

Open challenges

1. Need full set of Physics benchmarks to cover NAS program:
   $\rightarrow$Requires some community buy-in to implement the physics benchmarks
2. Getting community input for detector designs: need multiple detectors for unbiased comparison of performance
3. Compiling all the metrics for the optimization
4. Need dedicated algorithm design and optimization to fully establish performance on key physics measurements.

Open challenges (2)

- Distributed data storage (globus not sufficient) →Rucio?
- Need autogenerated monitoring webpage to present job artifacts via web-api (as opposed to browsing the job artifacts manually).
- Dispatch to HPC →Parsl, gitlab runner on dedicated servers, or kubernetes clusters

Summary

- Full simulation with parameterized detectors is possible
- Parameterization allows for optimization of design for physics performance.
- Identified new workflows which significantly lowers the barrier for entry.
- R&D prototype implementation promising, pattern could be repeated and scaled up for the EIC

Thank You !

Software

Primary Toolkit

- **DD4hep** – Detector description
- **Acts** – A Common Tracking Software
- **PODIO** – Data model tool

Browser workflow example

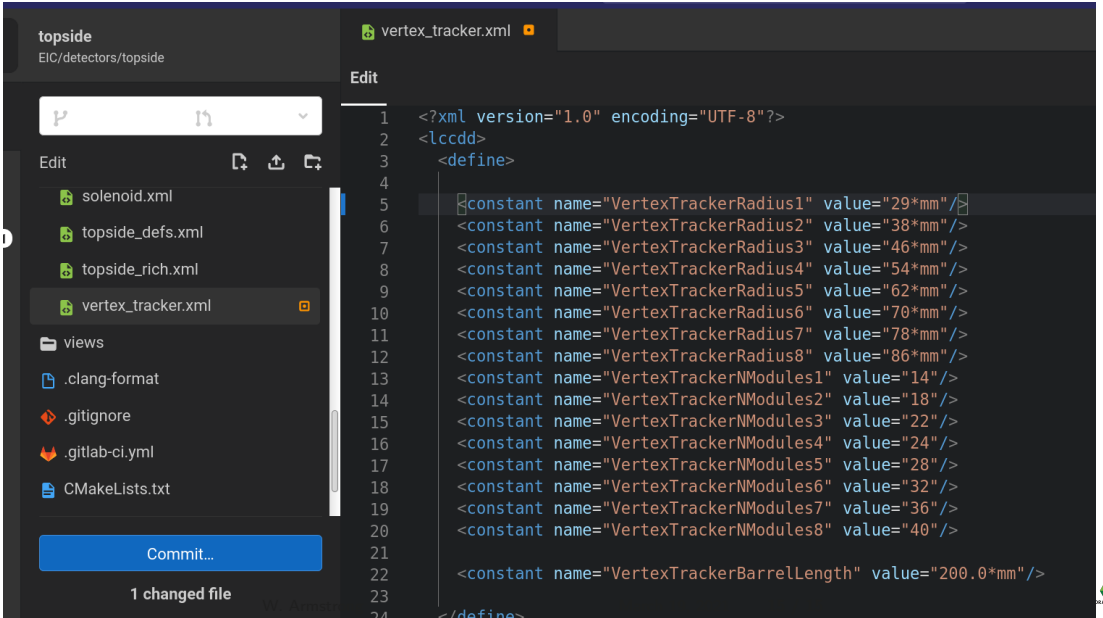
vertex_tracker.xml 8.8 KB

Edit | Web IDE | Replace | Delete

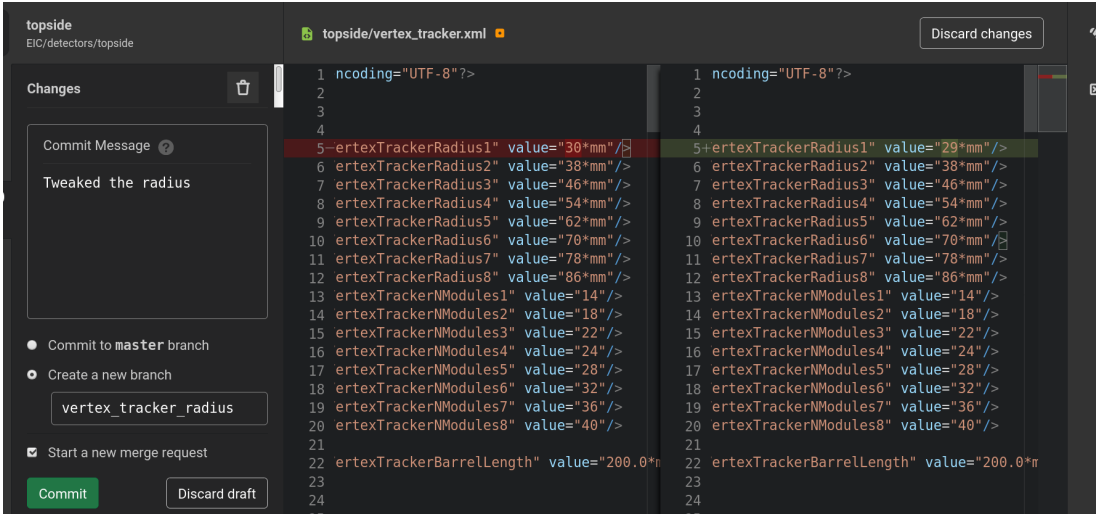
```xml
<?xml version="1.0" encoding="UTF-8"?>
<lccdd>
  <define>

    <constant name="VertexTrackerRadius1" value="30*mm"/>
    <constant name="VertexTrackerRadius2" value="38*mm"/>
    <constant name="VertexTrackerRadius3" value="46*mm"/>
    <constant name="VertexTrackerRadius4" value="54*mm"/>
    <constant name="VertexTrackerRadius5" value="62*mm"/>
    <constant name="VertexTrackerRadius6" value="70*mm"/>
    <constant name="VertexTrackerRadius7" value="78*mm"/>
    <constant name="VertexTrackerRadius8" value="86*mm"/>
    <constant name="VertexTrackerNModules1" value="14"/>
    <constant name="VertexTrackerNModules2" value="18"/>
    <constant name="VertexTrackerNModules3" value="22"/>
    <constant name="VertexTrackerNModules4" value="24"/>
    <constant name="VertexTrackerNModules5" value="28"/>
    <constant name="VertexTrackerNModules6" value="32"/>
```

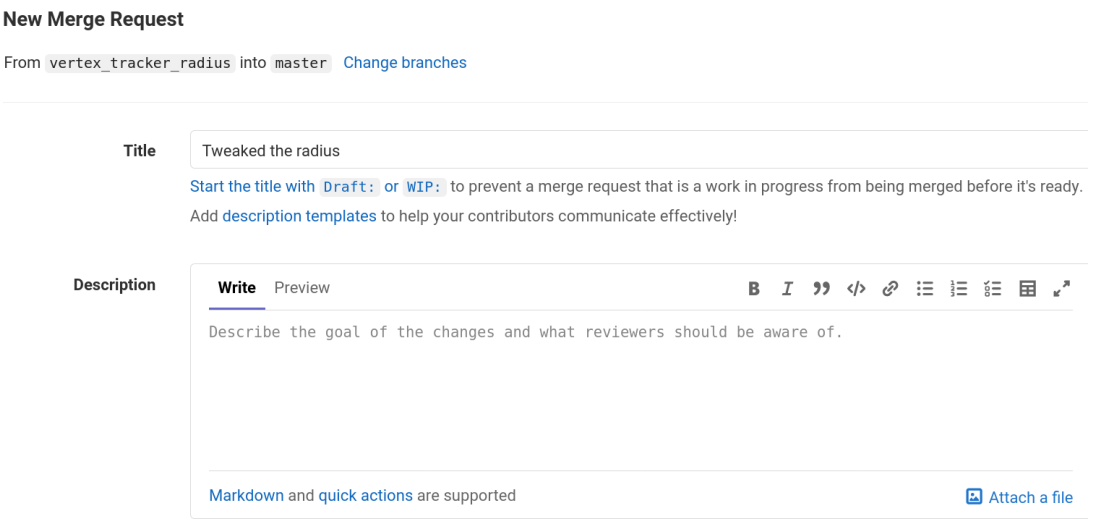
Browser workflow example



Browser workflow example



Browser workflow example



Browser workflow example

EIC › detectors › topside › Merge Requests › !91

Open Opened just now by Whitney Armstrong Owner Edit Mark as draft

Tweaked the radius

Overview 0 Commits 1 Changes 1

Request to merge vertex_tracker_ra... into master

Open in Web IDE Check out branch

Detached merge request pipeline #7395 running for 30bc4360

Browser workflow example