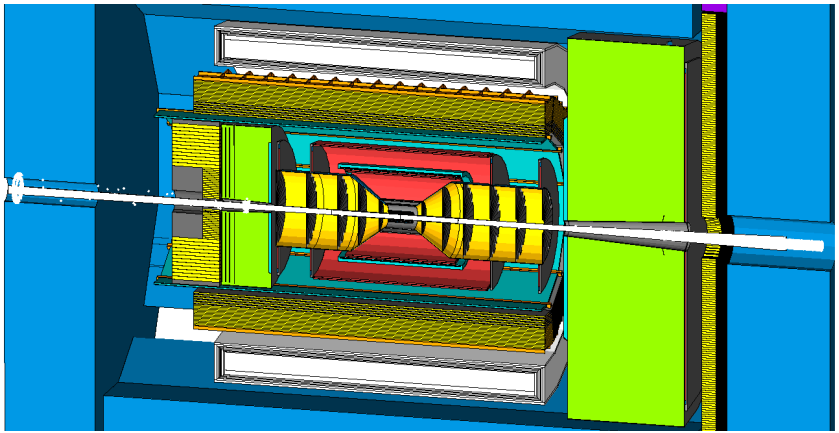
INTRODUCTION TO ATHENA SOFTWARE



CHAO PENG
Argonne National Laboratory

For Athena Software Working Group

02/22/2022

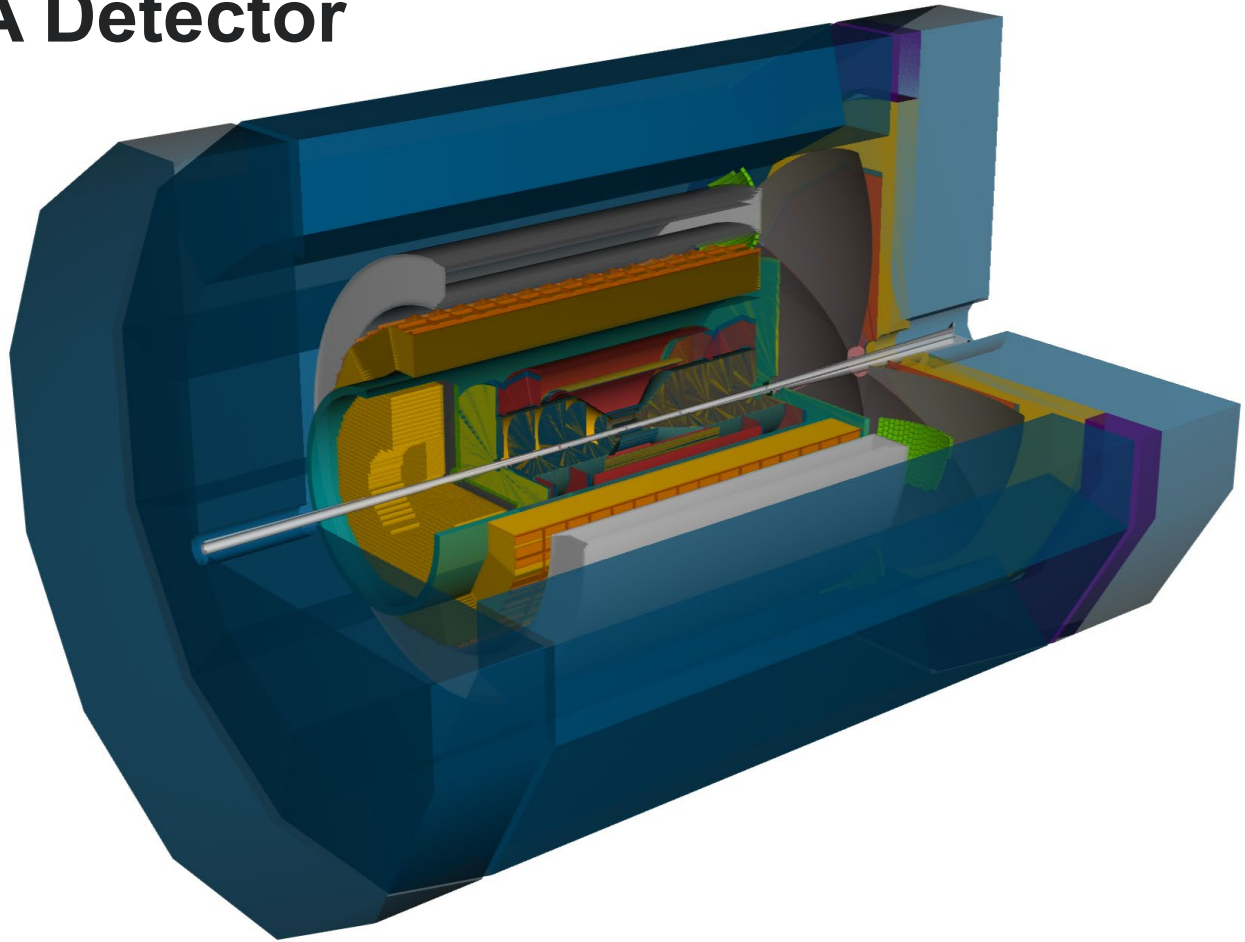
ATHENA Software

- Simulation with DD4hep https://eicweb.phy.anl.gov/EIC/detectors/athena
  - Subdetector plugins for Athena detector

- Data model: EICD https://eic.phy.anl.gov/eicd
  - Built upon PODIO
  - Flat data structure, connect tracks, hits, clusters with indices

- Analysis framework: Juggler https://eicweb.phy.anl.gov/EIC/juggler
  - Built upon GAUDI framework
  - Algorithms and tools for digitization, clustering, tracking, …

ATHENA Detector



SoLID with ATHENA Software

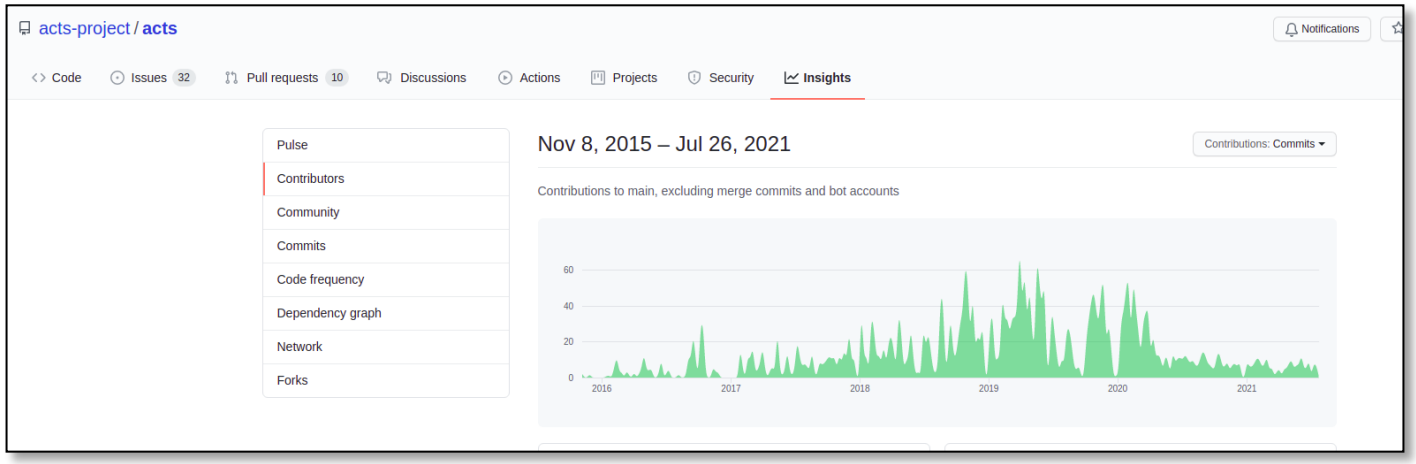
- It is a general software framework, so many pieces in it can be reused easily for SoLID

- What's in there
  - Working digitization, tracking, clustering, and so on
  - Many detector (tracker, calorimeter, RICH) plugins that can be modified easily
  - A SoLID repository with its major components (but unmaintained for a year) https://eicweb.phy.anl.gov/EIC/detectors/solid

- What's needed
  - Developing detailed detector plugins for SoLID
  - Optimizing algorithms for SoLID
  - Setup working benchmarks
  - All these works can be built upon existing ones

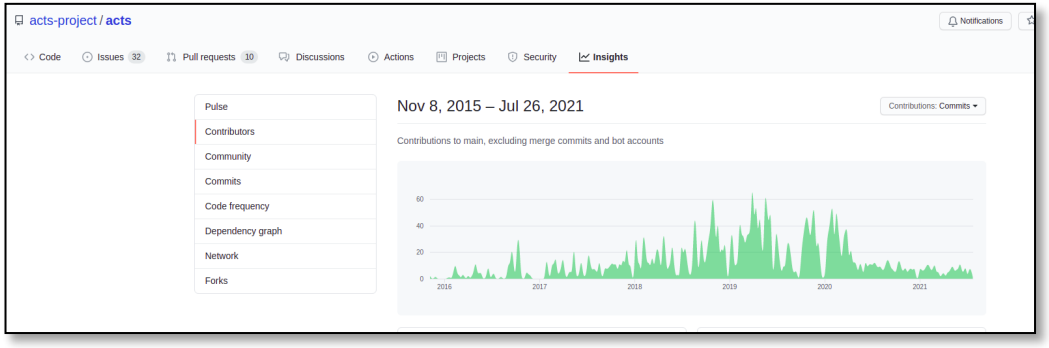
Acts for Tracking

- Acts is a particle track reconstruction toolkit that is widely used in high energy physics
  - Common algorithms for track propagation and fitting, seed finding, and vertexing
  - Independent of tracking detectors
  - Actively developing and well maintained
  - Native support for geometry description using DD4hep

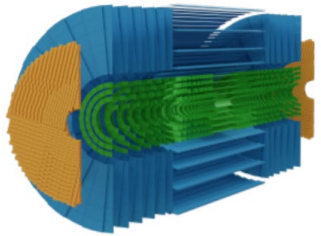


Acts for Tracking

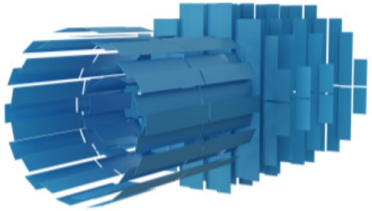
- Acts is a particle track reconstruction toolkit that is widely used in high energy physics
  - Common algorithms for track propagation and fitting, seed finding, and vertexing
  - Independent of tracking detectors
  - Actively developing and well maintained
  - Native support for geometry description using DD4hep



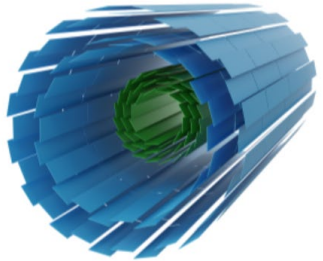
ATLAS ITk



PANDA silicon detector



sPHENIX silicon trackers



Tracking Benchmarks

- Using barrel trackers of the current Athena implementation
  https://eicweb.phy.anl.gov/EIC/benchmarks/reconstruction_benchmarks/-/tree/master/benchmarks/tracking
  - Also check track_finding, track_fitting benchmarks
- run_tracking_benchmarks.py

Simulation (particle gun)

```
if 'sim' in procs:
    # generate particles
    gen_cmd = ['python', gen_script, gen_file,
            '-n', '{}'.format(args.nev),
            '-s', '{}'.format(args.seed),
            '--etamin', '{}'.format(args.etamin), '--etamax', '{}'.format(args.etamax),
            '--pmin', '{}'.format(args.pmin), '--pmax', '{}'.format(args.pmax),
            '--particles', args.particles]
    subprocess.run(gen_cmd)
    # simulation
    sim_cmd = ['npsim',
            '--part.minimalKineticEnergy', '1*TeV',
            '--numberOfEvents', '{}'.format(args.nev),
            '--runType', 'batch',
            '--inputFiles', gen_file,
            '--outputFile', sim_file,
            '--compact', args.compact,
            '-v', 'WARNING']
    if args.seed > 0:
        sim_cmd += ['--random.seed', args.seed]
    return_code = subprocess.run(sim_cmd).returncode
    if return_code is not None and return_code < 0:
        print("ERROR running simulation!")
        exit(1)
    subprocess.run(['rootls', '-t', sim_file])
```

Reconstruction (digi. + tracking) and analysis

```
if 'rec' in procs:
    # export to environment variables (used to pass arguments to the option file)
    os.environ['JUGGLER_SIM_FILE'] = sim_file
    os.environ['JUGGLER_REC_FILE'] = rec_file
    os.environ['JUGGLER_COMPACT_PATH'] = args.compact
    os.environ['JUGGLER_N_EVENTS'] = '{}'.format(args.nev)

    juggler_xenv = os.path.join(os.environ.get('JUGGLER_INSTALL_PREFIX', '../local'), 'Juggler.xenv')

    rec_cmd = ['xenv', '-x', juggler_xenv, 'gaudirun.py', os.path.join(sdir, 'options', option_script)]
    return_code = subprocess.run(rec_cmd).returncode
    if return_code is not None and return_code < 0:
        print('ERROR running juggler ({})!'.format(opt))
        exit(1)
    process = subprocess.run(['rootls', '-t', rec_file])


if 'ana' in procs:
    os.makedirs('results', exist_ok=True)
    ana_cmd = ['python', analysis_script, rec_file,
               '--mc-collection', 'mcparticles2',
               '--tracking-collection', 'outputTrackParameters',
               '-o', 'results']
    return_code = subprocess.run(ana_cmd).returncode
    if return_code is not None and return_code < 0:
        print('ERROR running analysis ({})!'.format(ana))
        exit(1)
```

Tracking Benchmarks

- Reconstruction option file
  - call algorithms developed in Juggler
  - options/truth_seeded_tracking.py

- **Digitize: simulation hits -> readout signals.** noise, resolution smearing, time jitters could be added here.
- **Reconstruct: Readout signals -> hits** Only readout unit info is available here (position, signal strength, timing). “calibration” could be implemented here.

```
48  trk_b_digi = TrackerDigi("trk_b_digi",
49          inputHitCollection="TrackerBarrelHits",
50          outputHitCollection="TrackerBarrelRawHits",
51          timeResolution=8)
```

```
68  trk_b_reco = TrackerReco("trk_b_reco",
69          inputHitCollection = trk_b_digi.outputHitCollection,
70          outputHitCollection="TrackerBarrelRecHits")
```

- **Source link:** prepare data to feed Acts. Link measurements (rec_hits) to Acts surfaces (geometry information)
- **Seeding:** truth seeding from MC particles Other seeding algorithms from Acts are migrated but not thoroughly tested yet.

```
83
84  sourcelinker = TrackerSourcesLinker("trk_srcslnkr",
85          inputHitCollections=["VertexBarrelRecHits", "TrackerBarrelRecHits"],
86          outputSourceLinks="TrackerSourceLinks",
87          outputMeasurements="TrackerMeasurements",
88          OutputLevel=DEBUG)
89
90  ## Track param init
91  truth_trk_init = TrackParamTruthInit("truth_trk_init",
92          inputMCParticles="mcparticles",
93          outputInitialTrackParameters="InitTrackParams",
94          OutputLevel=DEBUG)
```

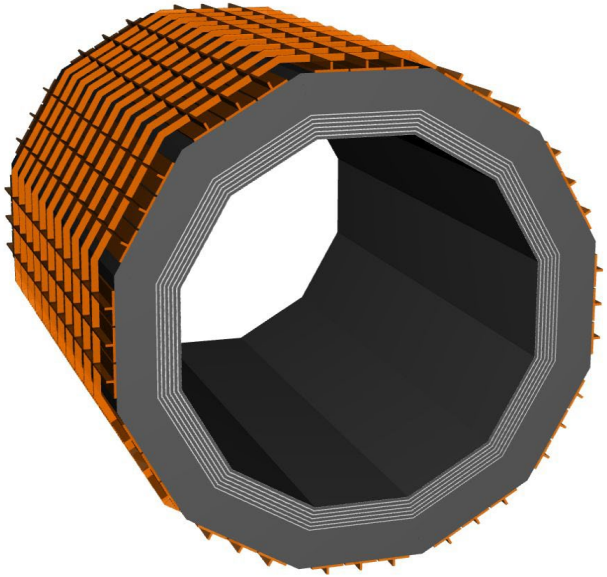
- **Tracking: Track finding and fitting with CKF** Combinatorial Kalman Filter from Acts.

```
96  # Tracking algorithms
97  trk_find_alg = TrackFindingAlgorithm("trk_find_alg",
98          inputSourceLinks = sourcelinker.outputSourceLinks,
99          inputMeasurements = sourcelinker.outputMeasurements,
100         inputInitialTrackParameters= truth_trk_init.outputInitialTrackParameters,
101         outputTrajectories="trajectories",
102         OutputLevel=DEBUG)
```

Clustering for Calorimeters

- In the analysis framework, clustering is done by
  - Digitization -> simulation hits to digitized signals
  - Hits reconstruction -> readout signals to energy/timing/position/etc
  - Proto-clustering -> grouping neighboring hits
  - Clustering reconstruction -> reconstruct energy/timing/position from group of hits

- Two clustering algorithms available, more developing
  - Island clustering for 2D hits
  - Topo clustering for 3D hits



Digitization/Reconstruction

- Simulate the electronics response. Currently in the benchmark:
  - ADC channels with specific dynamic range
  - Pedestal (mean + sigma)
  - Timing Jitter

  *ADC_value = hit_energy / dynamic_range * ADC_capacity + pedestal_mean + pedestal_error*
  (ADC_value clamped by [0, ADC_capacity])

  *Timing_value = (hit_time + time_jitter) * time_conversion_factor*

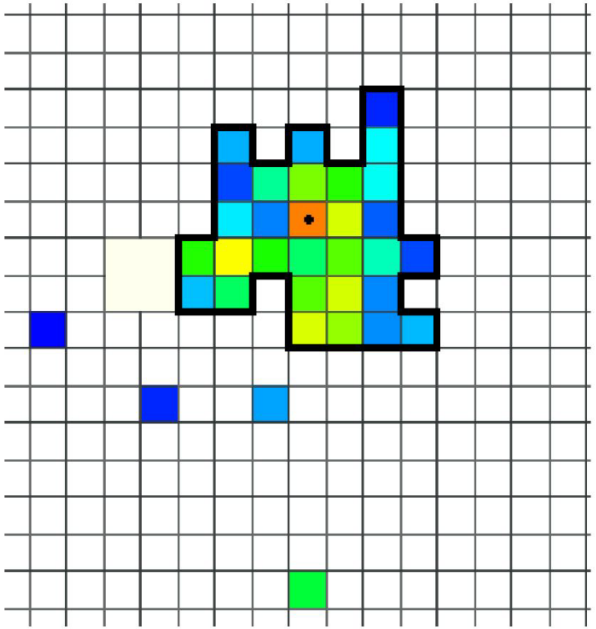
- Hits reconstruction is the reversed process without knowledge of pedestal error and time jitter (assume 0)

Island Clustering - Grouping

Group all neighbouring hits

- Parameterized conditions for finding neighbors
- Distance in local-XY, local-XZ, local-YZ,
- local-XY scaled by cell dimensions,
- global eta-phi, global R-phi

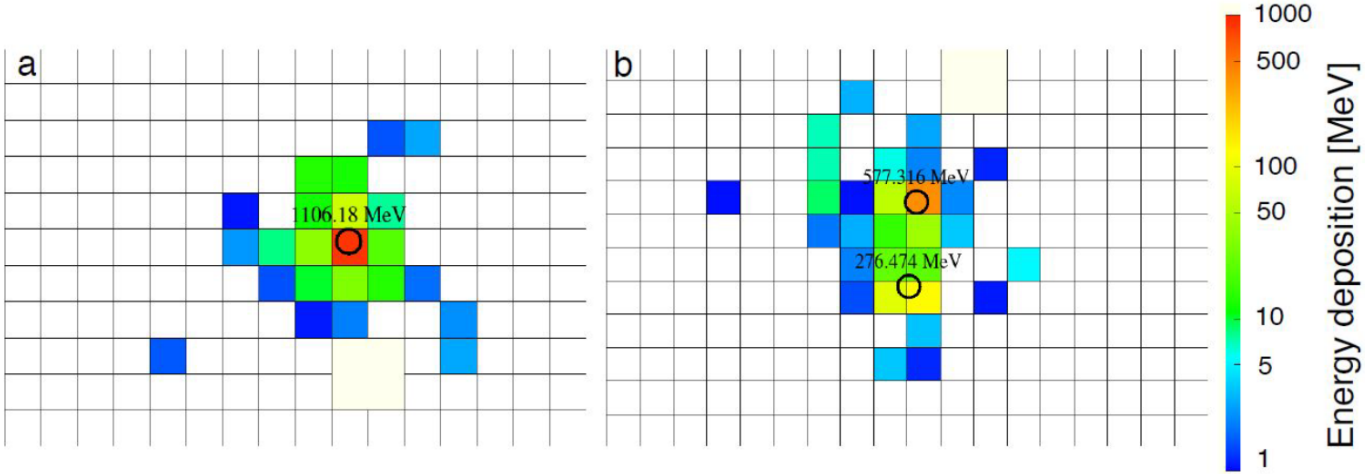
Parameterised minimal energy to be qualified as cluster center, and minimal energy to participate clustering



Island Clustering - Splitting

Cluster splitting is available for Island Clustering

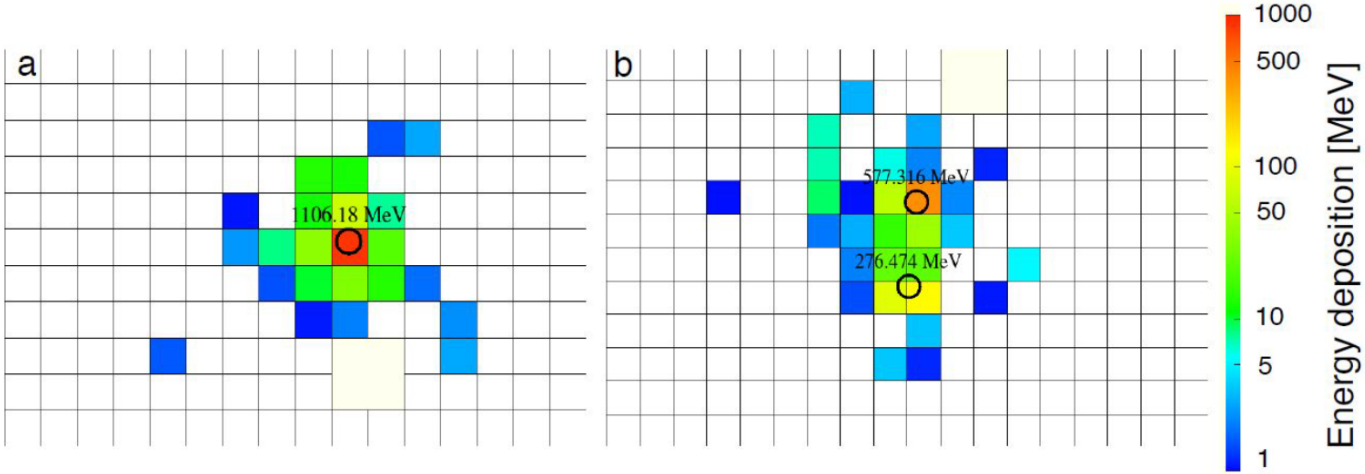
- Split based on Local maxima that are qualified as cluster center
- Hits energy split based on local maxima's energies and distances



Island Clustering - Splitting

Cluster splitting is available for Island Clustering

- Split based on Local maxima that are qualified as cluster center
- Hits energy split based on local maxima's energies and distances



Topo Clustering

Similar to Island clustering but works for hits from several layers, currently used for imaging layers

Hits at the same layer, local-XY

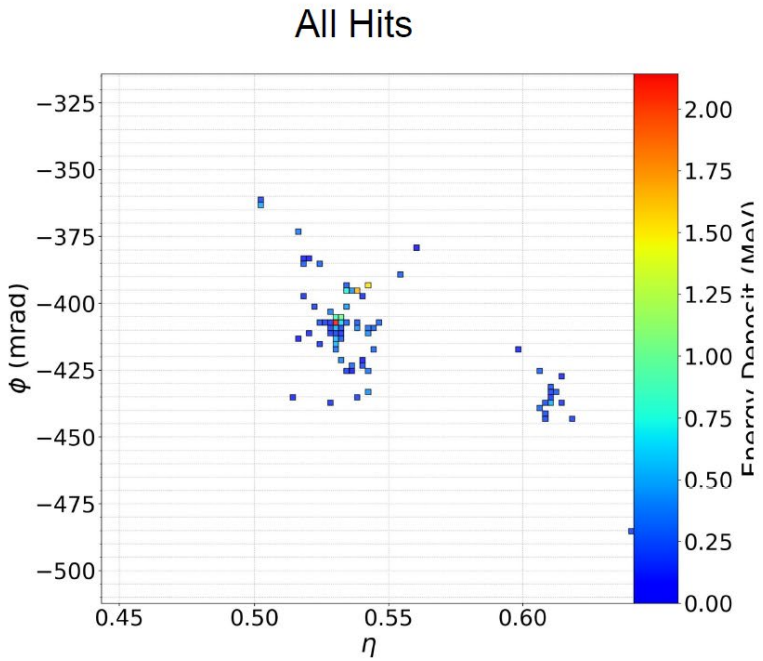
Hits from different layers, layer id difference and global eta-phi

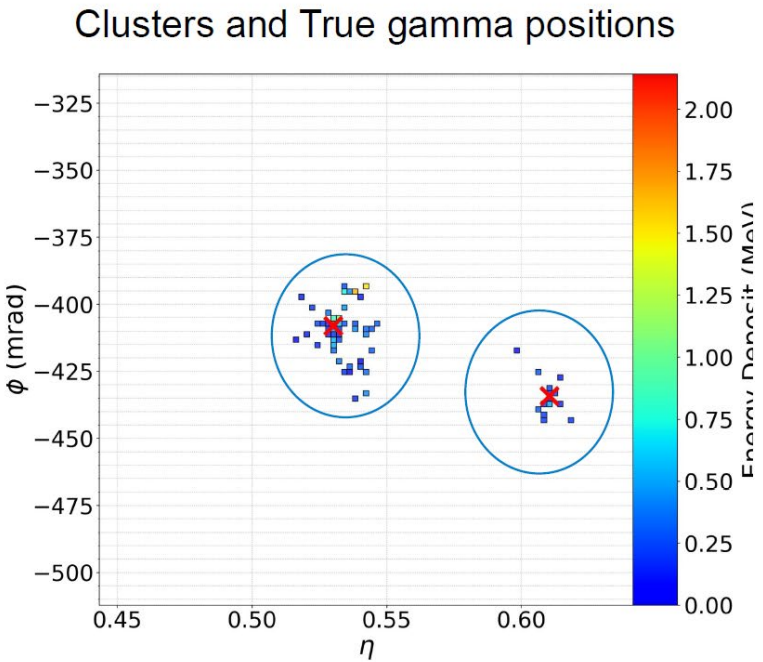
Hits from different sectors, global distance

No splitting implemented currently

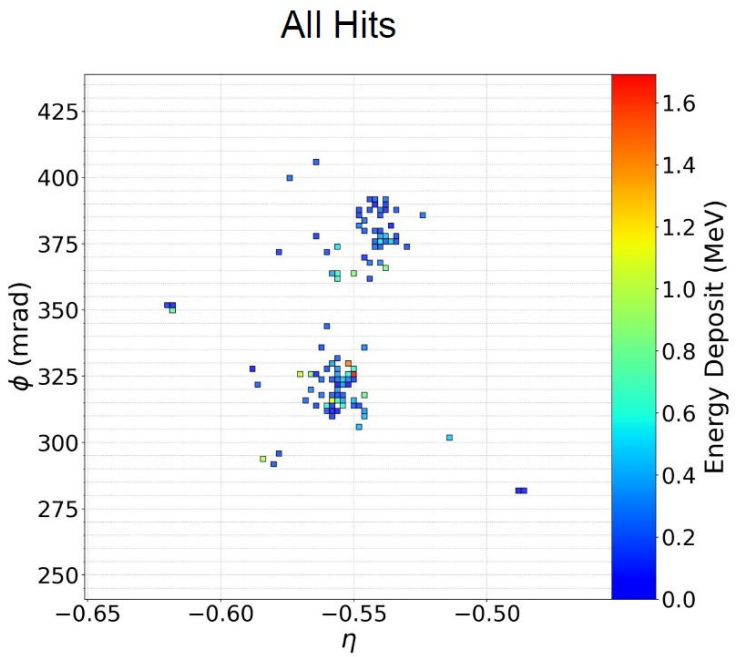
- Mostly MIP signals in imaging pixels

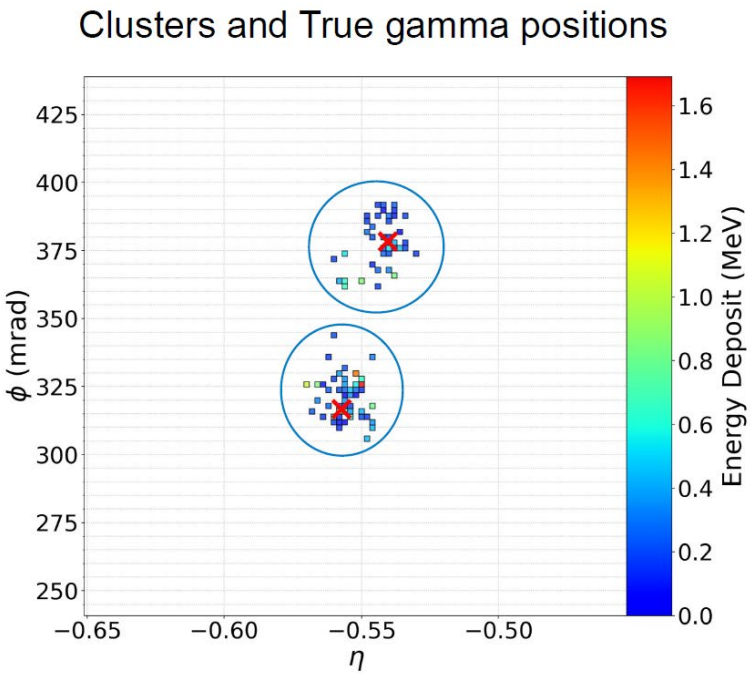
Example – 3D Clustering





Example – 3D Clustering





Example – Run Benchmarks

- **Run the example**

  Install EIC container

```
mkdir $HOME/eic && cd $HOME/eic
curl https://eicweb.phy.anl.gov/containers/eic_container/-/raw/master/install.sh | bash
```

  Run EIC container

```
./eic-shell
```

  Get Reconstruction benchmarks

```
git clone https://eicweb.phy.anl.gov/EIC/benchmarks/reconstruction_benchmarks.git
cd reconstruction_benchmarks
```

  Setup environment variables needed by the run script

```
source /opt/detector/setup.sh
export DETECTOR_PATH=/opt/detector/share/athena
export JUGGLER_DETECTOR=athena
export JUGGLER_INSTALL_PREFIX=/usr/local
```

  Run benchmark

```
python benchmarks/tracking/run_tracking_benchmarks.py --etamin=-3 --etamax=3 -n 100
```

Developing

- **Develop based on the example (after “run the example”, assumed in container)**

  Install Athena detector

```
cd $HOME/eic
git clone https://eicweb.phy.anl.gov/EIC/detectors/athena.git && cd athena
mkdir build && cd build
cmake .. -DCMAKE_INSTALL_PREFIX=$ATHENA_PREFIX
make -j install
export DETECTOR_PATH=$ATHENA_PREFIX/share/athena

cd $HOME/eic
git clone https://eicweb.phy.anl.gov/EIC/detectors/ip6.git && cd ip6
mkdir build && cd build
cmake .. -DCMAKE_INSTALL_PREFIX=$ATHENA_PREFIX
make -j install
cp –r ../ip6 $DETECTOR_PATH/
```

  Modify/adding detector

  See software tutorial
  https://eic.phy.anl.gov/tutorials/eic_tutorial/part1/simple_detector
  https://eic.phy.anl.gov/tutorials/eic_tutorial/part2/adding_detectors

Developing

- **Develop based on the example (after “run the example”)**

  Install Juggler

```
cd $HOME/eic
git clone https://eicweb.phy.anl.gov/EIC/juggler.git && cd juggler && git checkout v1.8.0
mkdir build && cd build
cmake .. -DCMAKE_INSTALL_PREFIX=$ATHENA_PREFIX
make -j install
export JUGGLER_INSTALL_PREFIX=$ATHENA_PREFIX
```

  Modify/adding algorithms

  See software tutorial
  https://eic.phy.anl.gov/tutorials/eic_tutorial/part3/running_juggler

  Run benchmark with modified detector and Juggler

```
cd $HOME/eic/reconstruction_benchmarks
python benchmarks/tracking/run_tracking_benchmarks.py --etamin=-3 --etamax=3 -n 100
```