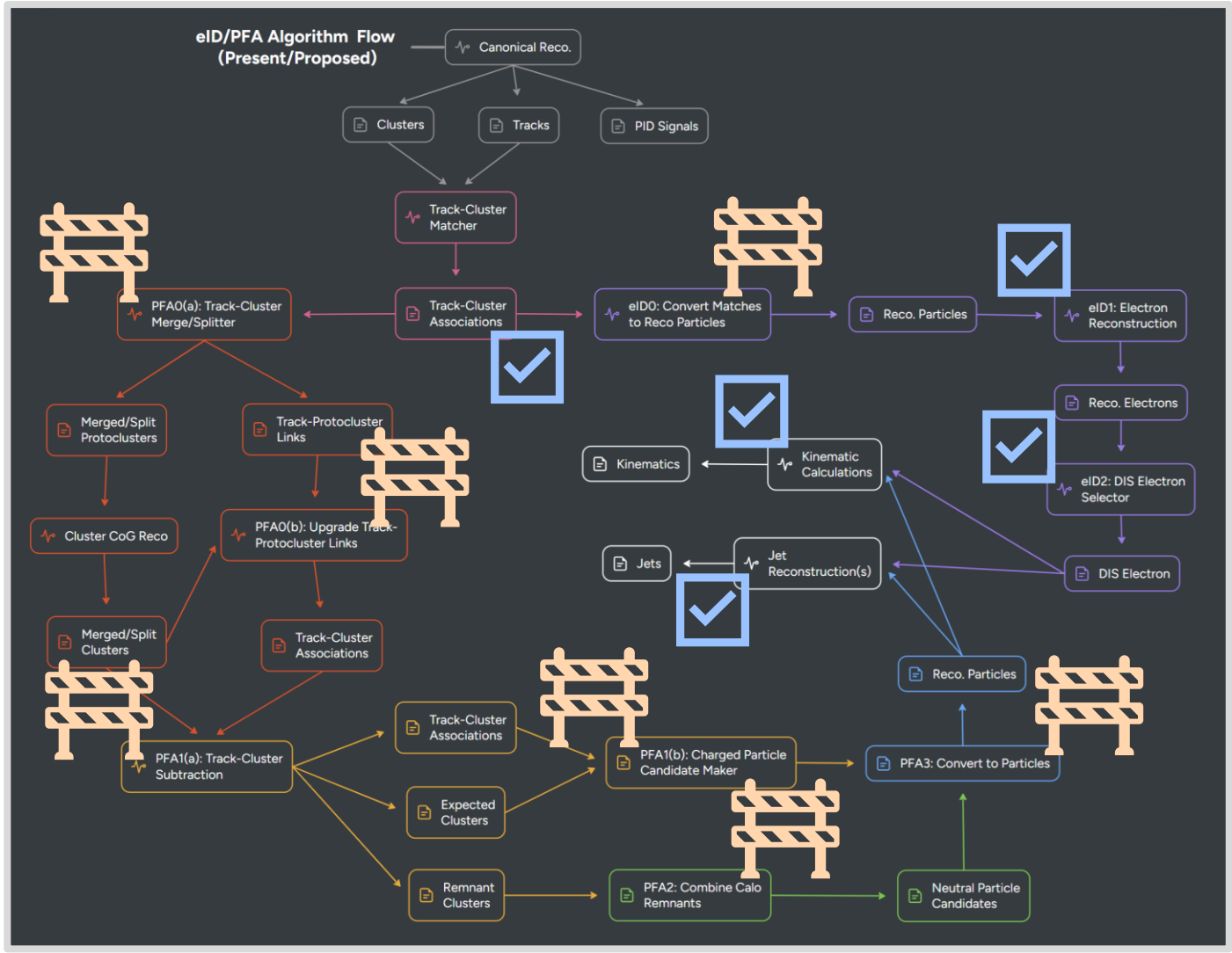
# PF Benchmarking | Roadmap & Tracking

- **Right:** Current PF roadmap
  - Initial dev phase is winding down, (almost) all should be in EICrecon by 26.04.0 campaign
  - Then we need to **tune & benchmark algorithms**
  - ∴ Now is a good time to start getting ready!

- **Note:** Spreadsheet for tracking tasks can be found here
  - 👍 Huge thanks to Shujie for sharing the template!

◯ = To-do
🚧 = In progress
☑ = Done/already in EICrecon
⊗ = Blocked



# PF Benchmarking | What Will You Be Doing?

- **Goal:** each stage should have a corresponding benchmark, **a ROOT macro/Python script which plots key metrics**
  - More details on metrics in following slides
  - Eventually will merge them into detector-benchmarks or physics-benchmarks
    - ☞ (Both also have lots of good examples!)
  - **But 1st target should be code that does what we need!**

- Will also use these macros/scripts to **tune parameters of each algorithm**
  - 26.04.0 campaign will use dummy values, so need to identify meaningful values
  - 2 additional metrics we'll be optimizing against:
    - **a) Jet energy scale/resolution**
    - **b) e- scale/resolution**
    - ☞ (Code exists for both of these already!)

- **Note:** can start testing code on output of algorithms even before we merge them
  - To checkout a branch:

    git clone git@github.com:eic/EICrecon.git
    git checkout –B **<branch-for-algo-pr>**
    **<compile eicrecon as usual>**
    source bin/eicrecon-this.sh
    **<run your sims/reco as usual>**
  - Links to PRs on following slides

- If you haven't checked them out yet, **these are good resources!**
  - General analysis tutorial
    - ☞ Esp. note *Full Chain Analysis* section and section on RDataFrames!
  - Snippets collects a lot of user-made examples
  - edm4eic.yaml, which defines (almost) everything in our data model
    - ☞ If not here, then it's in edm4hep.yaml

# PF Benchmarking | Algorithms & Links to PRs

| Task | Description | Issue/PR | Exp. Campaign |
|---|---|---|---|
| **PFA-1/eID3** | Deprecate MatchClusters, replace w/ pure reco equivalent | EICrecon#1956 | **26.05.0** |
| **PFA0(a)** | Complete merge/splitter update | EICrecon#1699 | **26.04.0** |
| **PFA0(b)** | Implement track-protocluster link promotion algorithm | EICrecon#2293 | **26.04.0** |
| **PFA1(a)** | Revive and finish track-cluster subtractor | EICrecon#1627 | **26.04.0** |
| **PFA1(b)** | Track-cluster converter (synergy w/ PFA-1) | EICrecon#2124 | **26.04.0** |
| **PFA2** | Implement calo remnant combiner | EICrecon#2195 | **26.04.0** |
| **PFA3** | Implement particle regressor/convertor | EICrecon#2399 | **26.04.0** |

- **Above:** algorithms in PFAlpha (excl. PFA-1).
  - The links for PFA0 – 3 point to the relevant PR for each
  - This means we need at least **4 benchmark macros/scripts!** (PFA0, 1, 2, 3)

# PF Benchmarking | Benchmarking Tasks (1/2)

| Tasks | Issue/PR/Note | Exp. Campaign | Assignee |
|---|---|---|---|
| **PFA-1/eID3 Benchmark**<br>**- input:** Sum eClust, sum pTrk, nClust, nTrk, E/p matched clusters, sum eGenPar, eGenPar, nGenPar<br>**- output:** Sum eRecPar, eRecPar, ePar, nRecPar, nPar, PES/R of reco pars | **To-do** | **CY26.Q3** | **HELP** |
| **PFA0 Benchmark**<br>**- input:** Sum eClust, eClust, pTrk, nTrk, nClust, E/p matched clusters<br>**- output:** Sum eSMClust, eSMClust, nSMClust, E/p SM clust, dRct SM | **Some work done** | **CY26.Q3** | **HELP** |
| **PFA1 Benchmark**<br>**- input:** Sum eClust, eClust, sum pTrk, pTrk, nTrk, nClust, E/p matched clusters, sum pChrgPar, pChrgPar, nChrgPar<br>**- output (expected):** sum eEXClust, eEXClust, nEXClust, E/p EX clust, dRct EX<br>**- output (remnant):** sum eREClust, eREClust, nREClust<br>**- output:** sum eEXClust + eREClust | **To-do** | **CY26.Q3** | **HELP** |

- **Above:** table of benchmarking tasks
  - Goal is to have ***tuning*** done by the end of Q3 this year (so the 26.10.0 campaign at latest)
  - Ideally benchmarks will be merged ahead of then, but can come later
- **Notes:** small text lists some ideas about what might be good to plot
  - PES/R = Particle Energy Scale/Resolution
  - SM = Split/Merge, EX = Expected, RE = Remnant
  - dRct = distance b/n cluster & matched track

# PF Benchmarking | Benchmarking Tasks (1/2)

| Tasks | Issue/PR/Note | Exp. Campaign | Assignee |
|---|---|---|---|
| **PFA2 Benchmark**<br>**- input:** sum eREClust (EM, H), eREClust (EM, H), nREClust (EM, H), sum eNeuPar, eNeuPar, nNeuPar<br>**- output:** sum eRecPar, nRecPar | **To-do** | **CY26.Q3** | **HELP** |
| **PFA3:**<br>**- input:** Sum eClust, sum pTrk, nClust, nTrk, E/p matched clusters, sum eGenPar, eGenPar, nGenPar<br>**- output:** Sum eRecPar, eRecPar, ePar, nRecPar, nPar, PES/R of reco pars | **To-do** | **CY26.Q3** | **HELP** |
| **PHYS Benchmark:** JES/R | **To-do** | **CY26.Q3** | Dener, **HELP** |
| **PHYS Benchmark:** Jets<br>- E, mass, FFs (jt, z), Substructure (dRcst, angularity, EECs) | **To-do** | **CY26.Q3** | Dener, **HELP** |
| **PHYS Benchmark:** Events<br>- TEECs, NECs | **Some work done** | **CY26.Q3** | Derek, **HELP** |

- **Above:** table of benchmarking tasks
  - Goal is to have ***tuning*** done by the end of Q3 this year (so the 26.10.0 campaign at latest)
- **Notes:** small text lists some ideas about what might be good to plot
  - EM = “Electromagnetic”, H = “Hadronic
  - dRcst = constituent delta-R
  - PHYS benchmarks not on critical path

# Algorithms | PFA0 | Track-Cluster Merge/Splitter

- **Track-Cluster Merging/Splitting:** merges and then splits clusters based on track projections
  - Algorithm based on ATLAS's split recovery procedure
    - c.f. Eur. Phys. J. C (2017) 77:466
  - Implemented in EICrecon#1406, updating in EICrecon#1699

- **The algorithm:**
  1) Match track projection to cluster
  2) If matched, calculate significance b/n $E_{clust}$ energy & expected $E_{dep}$:

$$S(E_{clust}) = \frac{E_{clust} - (p_{proj} \times \langle E/p \rangle)}{\sigma(E_{dep})}$$

  3) If $S < S_{cut}$, add clusters inside $\Delta r_{add}$
  4) If multiple tracks pointing to merged cluster:
     3) Split into one cluster for each track & reweight transverse shape by $p_{trk}$, track projection

**Inputs:**
- Clusters
- Track projections (**to-do:** track-cluster matches)
- **Optional:** reco-sim hit associations (or sim hits)

**Outputs:**
- Merged/Split clusters
- Track-merged/split cluster matches
- **Optional:** scale for transverse shape reweighting

**Parameters:**
- $\langle E/p \rangle$: average E/p
- $\sigma(E_{dep})$: spread of dep. energy
- $S_{cut}$: threshold to run split-recovery
- $\Delta r_{add}$: window to add clusters
- $\sigma_{trk}$: scale for transverse shape reweighting

# Algorithms | PFA1(a) | Track-Cluster Subtractor (1/2)

- **Track-Cluster Subtractor:** subtracts momentum of matched track(s) from cluster
  - In progress at EICrecon#1627

- **The algorithm:**
  1) Build map of clusters onto *all* matched tracks
  2) For each cluster:
     a) Sum energy of matched tracks:

$$E_{trk} = \sum p_{trk}(S_{use}) \oplus m_{trk}$$

     b) Subtract sum: $E_{sub} = E_{clust} - f_{sub}E_{trk}$
     c) If NOT consistent w/ 0,
        - Create remnant cluster w/ $E_{sub}$
        - Set subtracted cluster energy to $E_{clust} - E_{sub}$
     d) Create an association for each track matched to subtracted cluster

**Inputs:**
- Track-cluster matches
- Track projections

**Outputs:**
- Subtracted clusters ($\sim E_{clust}$)
- Remnant clusters ($E_{clust} - E_{trk}$)
- Track-subtracted

**Parameters:**
- $f_{sub}$: fraction of track energy to subtract
- $m_{default}$: default mass to use for track energy
- $S_{use}$: surface to evaluate track momentum at
- $k_{do\ n\sigma?}$: turn on/off checking against resolutions
- $n\sigma_{cut}$: max no. of sigmas to be consistent w/ 0
- $\sigma_{trk}$: tracking resolution to use in n-sigma cut
- $\sigma_{cal}$: calo resolution not use in n-sigma cut

# Algorithms | PFA1(a) | Track-Cluster Subtractor (2/2)

- **Track-Cluster Subtractor:** subtracts momentum of matched track(s) from cluster
  - In progress at EICrecon#1627

- **The algorithm:**
  1) Build map of clusters onto *all* matched tracks
  2) For each cluster:
     a) Sum energy of matched tracks:

$$E_{trk} = \sum p_{trk}(S_{use}) \oplus m_{trk}$$

     b) Subtract sum: $E_{sub} = E_{clust} - f_{sub}E_{trk}$
     c) If NOT consistent w/ 0,
        - Create remnant cluster w/ $E_{sub}$
        - Set subtracted cluster energy to $E_{clust} - E_{sub}$
     d) Create an association for each track matched to subtracted cluster

**Sub-routine:** is $E_{sub}$ consistent w/ zero?

1) If $E_{sub} < 0$, **YES**
2) Else if $k_{do\ n\sigma?}$
   a) Calculate $n\sigma$

$$n\sigma = \frac{E_{sub}}{\sigma_{trk} \oplus \sigma_{cal}}$$

   b) If $n\sigma < n\sigma_{cut}$, **YES**
3) Else
   a) If $E_{sub} < \epsilon$, **YES**

**Note:** epsilon here is std::numeric_limits<double>::epsilon()

# Algorithms | PFA1(b) | Charged Candidate Maker

- **Charged Candidate Maker:** forms track-cluster matches into a charged particle candidate
  - In progress at EICrecon#2124

- **The algorithm:**
  1) Build map of tracks onto ***all matched clusters***
  2) For each **track:**
     a) For each matched cluster:
        i. Identify if in an ECal or an HCal by checking system ID
        ii. Select relevant weight
        iii. Add to relevant members
     b) Add to relevant member

**Inputs:**
- Track-cluster matches

**Outputs:**
- Charged particle candidates

**Parameters:**
- None!

# Algorithms | PFA2 | Calo Remnant Combiner

- **Calo Remnant Combiner:** combines remnant clusters from subtractor into neutral particle candidates
  - In progress at EICrecon#2195

- **The algorithm:**
  1) Combine nearby ECal, HCal clusters
     a) Identify seed ECal cluster
     b) Merge all ECal, HCal clusters in $\Delta r_{add}^{em}$, $\Delta r_{add}^{h}$ of seed and create neutral candidate
     c) Repeat until no ECal clusters are left
  2) Combine remaining HCal clusters
     a) Identify seed HCal cluster
     b) Add all HCal clusters in $\Delta r_{add}^{h}$ of seed and create neutral candidate
     c) Repeat until no HCal clusters are left

**Inputs:**
- Remnant ECal clusters
- Remnant HCal clusters

**Outputs:**
- Neutral particle candidates

**Parameters:**
- $\Delta r_{add}^{em}$: window to add ECal clusters
- $\Delta r_{add}^{h}$: window to add HCal clusters

# Algorithms | PFA3 | Candidate Particle Convertor (1/2)

- **Particle Converter:** takes candidate particles and turns them into reconstructed particles
  - In progress at EICrecon#2399

- **The algorithm:**
  1) Assign preliminary PID based on what info is available (next slide)
  2) Calculate track energy
  $$E_{trk} = p_{trk} \oplus m_{pid}$$
  3) Calculate calorimeter energy
  $$E_{cal} = N_{cal}\left(\sum w_{em}E_{em} + \sum w_h E_h\right)$$
  4) If charged particle and $k_{use\ \sigma?}$, calculate resolution-weighted average of $E_{cal}$ and $E_{trk}$
  5) Calculate remaining kinematics and create reconstructed particle

**Inputs:**
- Candidate charged/neutral particles
- Reconstructed charged particles (for PID)
- Primary vertices (for neutral candidates)

**Outputs:**
- Reconstructed particles

**Parameters:**
- $k_{use\ \sigma?}$: turn on/off using resolution in energy calculation for charged candidates
- $N_{cal}$: normalization of calo energy
- $\sigma_{trk}$: tracking resolution to use in energy calc
- $\sigma_{cal}$: calo resolution to use in energy calc

# Algorithms | PFA3 | Candidate Particle Convertor (2/2)

- **Particle Converter:** takes candidate particles and turns them into reconstructed particles
  - In progress at EICrecon#2399

- **The algorithm:**
  1) Assign preliminary PID based on what info is available (next slide)
  2) Calculate track energy
  $$E_{trk} = p_{trk} \oplus m_{pid}$$
  3) Calculate calorimeter energy
  $$E_{cal} = N_{cal}\left(\sum w_{em}E_{em} + \sum w_h E_h\right)$$
  4) If charged particle and $k_{use\ \sigma?}$, calculate resolution-weighted average of $E_{cal}$ and $E_{trk}$
  5) Calculate remaining kinematics and create reconstructed particle

**Sub-routine:** what PDG to assign?

1) Check what info is present:
   a) **If** at least 1 related track, **hasTrk = TRUE**
   b) **If** at least 1 EMCal cluster, **hasECal = TRUE**
   c) **If** at least 1 HCal cluster, **hasHCal = TRUE**
2) **If hasTrk**
   a) Use track to retrieve reco charged particle
   b) PDG = chrgPar.getPDG()
3) **Else**
   a) **If hasECal and !hasHCal,** PDG = <Photon>
   b) **If hasECal and hasHCal,** PDG = <Neutron>
   c) **If !hasECal and hasHCal,** PDG = <???>

**Note:** for initial pass electron/muon and pi0/gamma discrimination deferred to downstream
   ☞ Also can refine neutral PID by checking relative ECal vs. HCal contribution