Key4hep native FastJet clustering algorithms

Jennifer Roloff

July 10, 2024

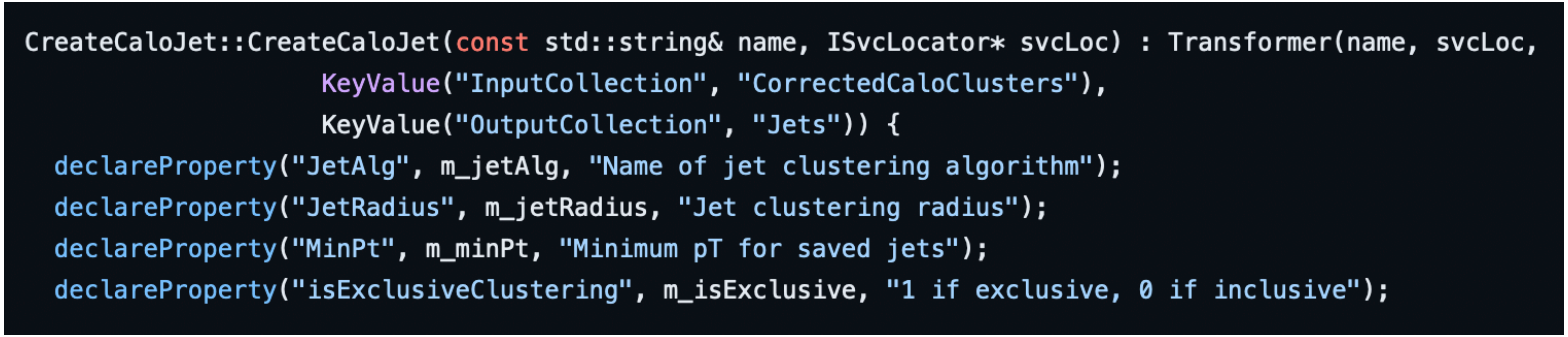
BROWN

Motivation

- Wanted to have standardized jet clustering algorithms available for testing
- Have created tools for calorimeter jet clustering and truth jet clustering
  - Can add one for particle flow and/or tracks as well
- Also created a tool for filtering truth particles to get stable particles
- See https://github.com/HEP-FCC/k4RecCalorimeter/pull/80 (merged) and https://github.com/HEP-FCC/k4RecCalorimeter/pull/95 (not yet merged) for more details on the code

Code structure and usage

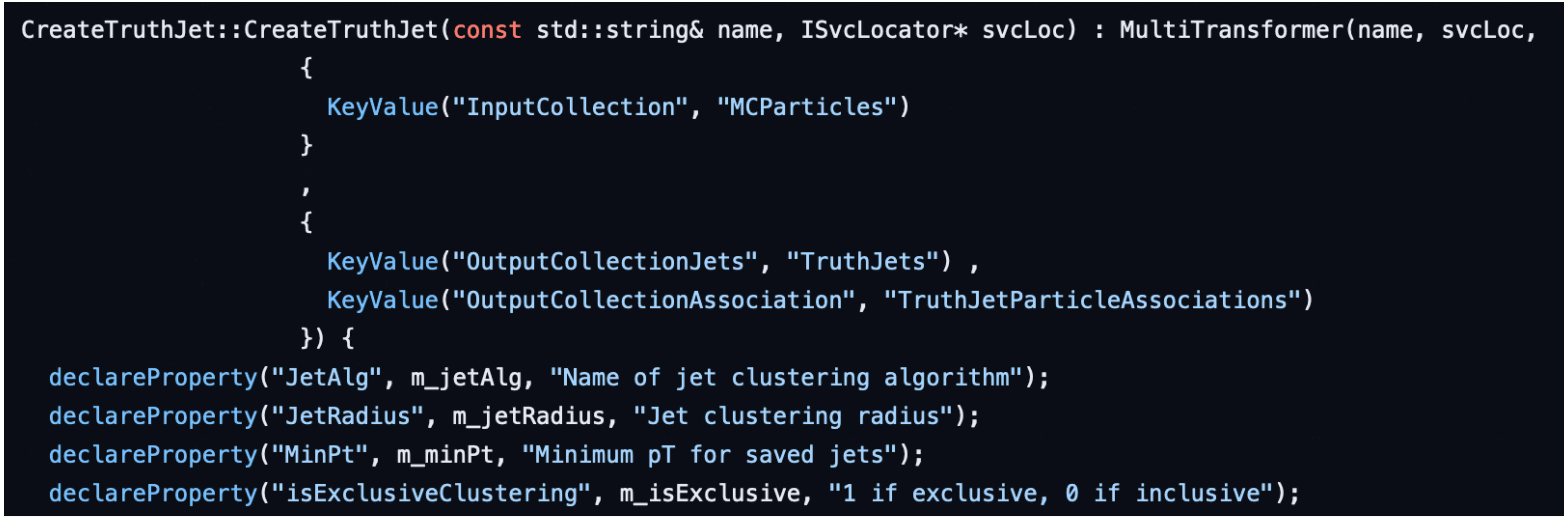
- Using Gaudi::(Multi)Transformer for jet reconstruction
  - Can configure the jet algorithm, jet radius, minimum p_T, and whether to use exclusive or inclusive clustering
    - Need to add more options for exclusive clustering (e.g. number of jets)
  - Can specify the output jet collection (edm4hep::ReconstructedParticleCollection)
  - For calorimeter jets, can specify the input collection (edm4hep::ClusterCollection)



```
CreateCaloJet::CreateCaloJet(const std::string& name, ISvcLocator* svcLoc) : Transformer(name, svcLoc,
                    KeyValue("InputCollection", "CorrectedCaloClusters"),
                    KeyValue("OutputCollection", "Jets")) {
  declareProperty("JetAlg", m_jetAlg, "Name of jet clustering algorithm");
  declareProperty("JetRadius", m_jetRadius, "Jet clustering radius");
  declareProperty("MinPt", m_minPt, "Minimum pT for saved jets");
  declareProperty("isExclusiveClustering", m_isExclusive, "1 if exclusive, 0 if inclusive");
```

Code structure and usage

- Using Gaudi::(Multi)Transformer for jet reconstruction — mostly the same configuration options as for calorimeter jets
  - Can also specify the name of the collection that has links between jets and their truth particle constituents (more on this later)
  - For truth jets, can specify the edm4hep::MCRecoParticleCollection to use



```
CreateTruthJet::CreateTruthJet(const std::string& name, ISvcLocator* svcLoc) : MultiTransformer(name, svcLoc,
                    {
                      KeyValue("InputCollection", "MCParticles")
                    }
                    ,
                    {
                      KeyValue("OutputCollectionJets", "TruthJets") ,
                      KeyValue("OutputCollectionAssociation", "TruthJetParticleAssociations")
                    }) {
  declareProperty("JetAlg", m_jetAlg, "Name of jet clustering algorithm");
  declareProperty("JetRadius", m_jetRadius, "Jet clustering radius");
  declareProperty("MinPt", m_minPt, "Minimum pT for saved jets");
  declareProperty("isExclusiveClustering", m_isExclusive, "1 if exclusive, 0 if inclusive");
```

Accessing constituents

- Needed to use different strategies to store and access jet constituents
  - For calorimeter jets, can use “addToClusters” function from ReconstructedParticle
    - Use existing clusters, but adds a link to this collection
  - No similar function for adding truth particle, and want to minimize duplication of information
  - Adding a collection of associations between a ReconstructedParticle and an MCRecoParticle (one per constituent)
    - Also need to save this association collection → ongoing MR to use MultiTransformers to make this strategy thread safe
- See backup slide for details on accessing this information

Thanks!

Example code for reading jet constituents

```
from podio import root_io
podio_reader = root_io.Reader('output_fullCalo_SimAndDigi.root')

for event in podio_reader.get("events"):
    jets = event.get("Jets")
    for jet in jets:
      clusters = jet.getClusters()
      for cluster in clusters:
        print (cluster.getEnergy())

    jets = event.get("TruthJets")
    associationColl = event.get(“TruthJetsAssociations")

    for jet in jets:
     for assoc in associationColl:
        if (assoc.getRec() == jet) :
          print(assoc.getSim().getEnergy())
```