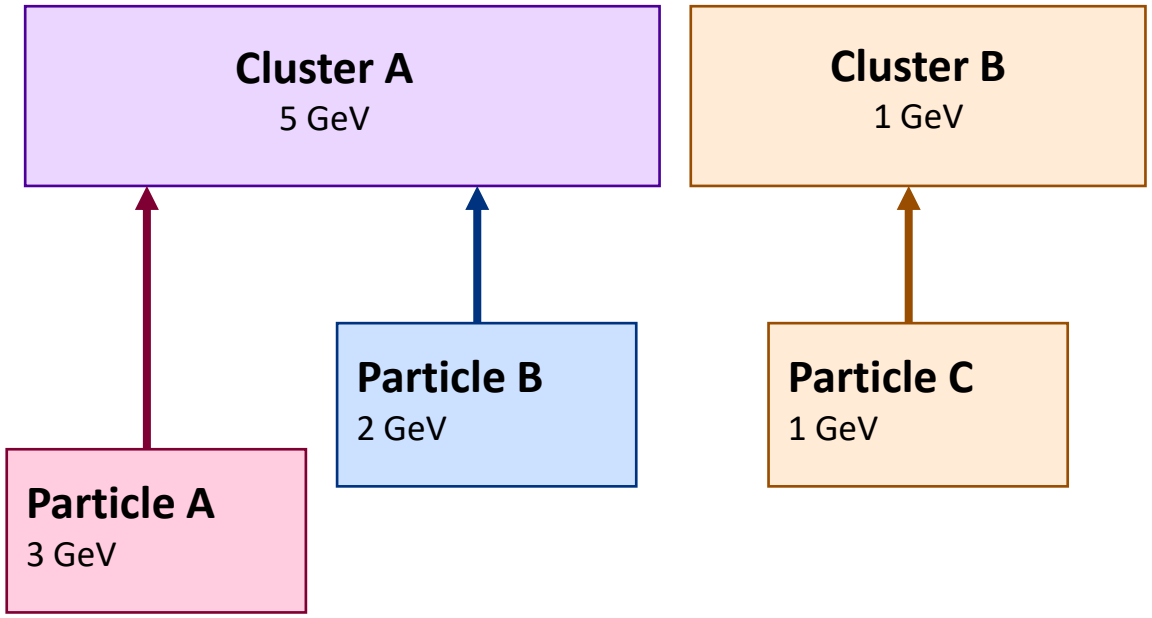
Truth Clustering | Reminder | Cluster Associations

- **Reminder:** clusters can have multiple MCParticle-Cluster associations (e.g. see the logic here)
  - **Below:** if particles A, B, C contributed listed energies to clusters A, B; then we would get 3 associations to right
  - Note that associations are made **between cluster and primary particles**



Association 1 | weight: 0.60

Cluster A (rec) — **Particle A** (sim)

Association 2 | weight: 0.40

Cluster A (rec) — **Particle B** (sim)

Association 3 | weight: 1.0

Cluster B (rec) — **Particle C** (sim)

Truth Clustering | The Issue (1/2)

- Association logic in place since Sep 2024
  - **However:** didn't update truth clustering at the time
  - ☞ It still assigns sim hits to particles based on the 1st contribution in vector

- Also introduced associations between raw hits and sim hits in Aug 2024
  - **Before:** raw-sim hits matched based on Cell ID
  - ☞ This is still done in truth clustering, though

∴ Truth clusters out-of-date in 2 regards:
  a) Particle assignment at odds w/ MC-Cluster Associations
  b) Does not use raw-sim hit relations

```
// Loop over all calorimeter hits and sort per mcparticle
for (const auto& hit : *hits) {
  // The original algorithm used the following to get the mcHit:
  //
  //    const auto& mcHit     = mc[hit->getObjectID().index];
  //
  // This assumes there is a one-to-one relation between the truth hit
  // (hits) and the reconstructed hit (mc). At least insofar as the
  // index in "hits" is being used to index the "mc" container.
  //
  // If the objects in "hits" have not been added to a collection,
  // then they will have getObjectID().index = "untracked" = -1
  //
  // The way we handle this is here is to check if getObjectID().index
  // is within the size limits of mc which includes >=0. If so, then
  // assume the old code is valid. If not, then we need to search
  // for the right hit.
  // FIXME: This is clearly not the right way to do this! Podio needs
  // FIXME: to be fixed so proper object tracking can be done without
  // FIXME: requiring Collection classes be used to manage all objects.
  std::size_t mcIndex = 0;
  if ((hit.getObjectID().index >= 0) &&
      (hit.getObjectID().index < static_cast<long>(mc->size()))) {
    mcIndex = hit.getObjectID().index;
  } else {
    mcIndex      = 0;
    bool success = false;
    for (auto tmpmc : *mc) {
      if (tmpmc.getCellID() == hit.getCellID()) {
        success = true;
        break;
      }
      mcIndex++;
    }
    if (not success) {
      continue; // ignore hit if we couldn't match it to truth hit
    }
  }

  const auto& trackID = (*mc)[mcIndex].getContributions(0).getParticle().getObjectID().index;
  // Create a new protocluster if we don't have one for this trackID
  if (!protoIndex.contains(trackID)) {
    clusters->create();
    protoIndex[trackID] = clusters->size() - 1;
  }
  // Add hit to the appropriate protocluster
  (*clusters)[protoIndex[trackID]].addToHits(hit);
  (*clusters)[protoIndex[trackID]].addToWeights(1);
}
}
```

Truth Clustering | The Issue (2/2)

∴ Truth clusters out-of-date in 2 regards:
- a) Particle assignment at odds w/ MC-Cluster Associations
- b) Does not use raw-sim hit relations

- Noted these points in issue EICrecon#2072
  - Dmitry R also ran into this in EICrecon#2323
  - Trying to address (b) in PR EICrecon#2336
    - **But:** need to figure out why change results in 5x as many EEEMCal clusters…

☞ **Today:** want to run some ideas past you all to address (a)

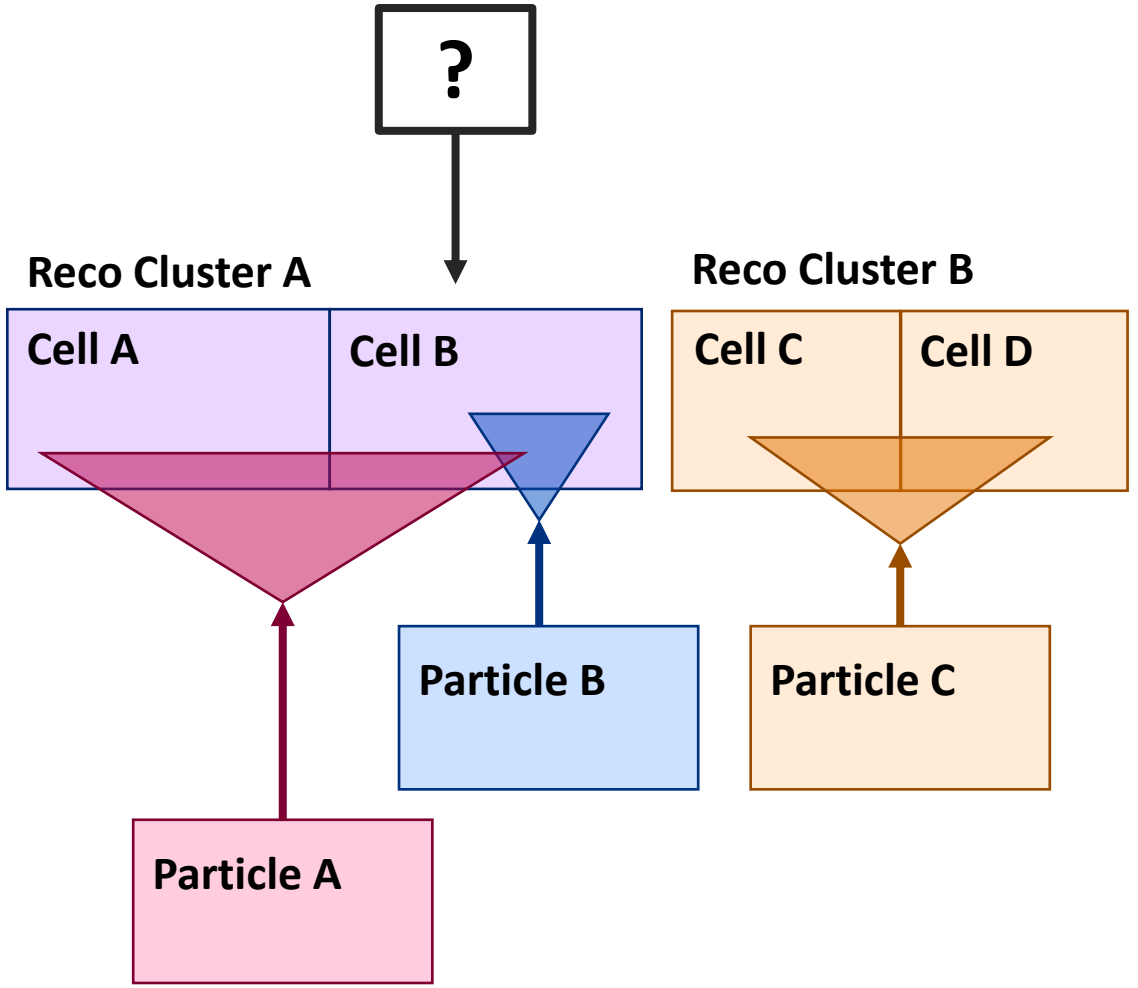
```
// Loop over all calorimeter hits and sort per mcparticle
for (const auto& hit : *hits) {
  // The original algorithm used the following to get the mcHit:
  //
  //    const auto& mcHit     = mc[hit->getObjectID().index];
  //
  // This assumes there is a one-to-one relation between the truth hit
  // (hits) and the reconstructed hit (mc). At least insofar as the
  // index in "hits" is being used to index the "mc" container.
  //
  // If the objects in "hits" have not been added to a collection,
  // then they will have getObjectID().index = "untracked" = -1
  //
  // The way we handle this is here is to check if getObjectID().index
  // is within the size limits of mc which includes >=0. If so, then
  // assume the old code is valid. If not, then we need to search
  // for the right hit.
  // FIXME: This is clearly not the right way to do this! Podio needs
  // FIXME: to be fixed so proper object tracking can be done without
  // FIXME: requiring Collection classes be used to manage all objects.
  std::size_t mcIndex = 0;
  if ((hit.getObjectID().index >= 0) &&
      (hit.getObjectID().index < static_cast<long>(mc->size()))) {
    mcIndex = hit.getObjectID().index;
  } else {
    mcIndex      = 0;
    bool success = false;
    for (auto tmpmc : *mc) {
      if (tmpmc.getCellID() == hit.getCellID()) {
        success = true;
        break;
      }
      mcIndex++;
    }
    if (not success) {
      continue; // ignore hit if we couldn't match it to truth hit
    }
  }

  const auto& trackID = (*mc)[mcIndex].getContributions(0).getParticle().getObjectID().index;
  // Create a new protocluster if we don't have one for this trackID
  if (!protoIndex.contains(trackID)) {
    clusters->create();
    protoIndex[trackID] = clusters->size() - 1;
  }
  // Add hit to the appropriate protocluster
  (*clusters)[protoIndex[trackID]].addToHits(hit);
  (*clusters)[protoIndex[trackID]].addToWeights(1);
}
}
```

Truth Clustering | Updating Assignment (1/3)

- Algorithm sorts reco hits by contributors*, creating a cluster for each unique contributor
  - * Based on the 1st contributor in relations…

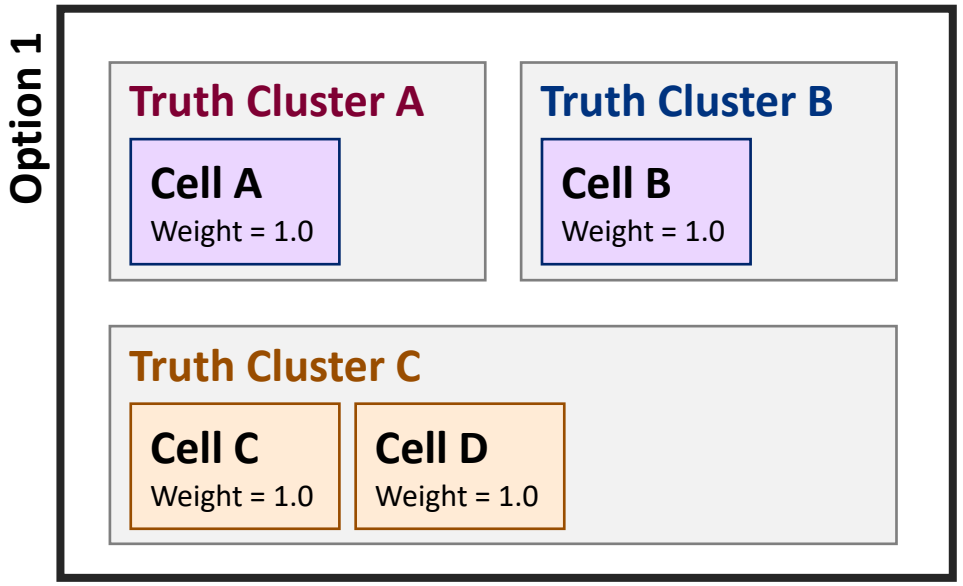
∴ Towards (a), there are 2 items to address:

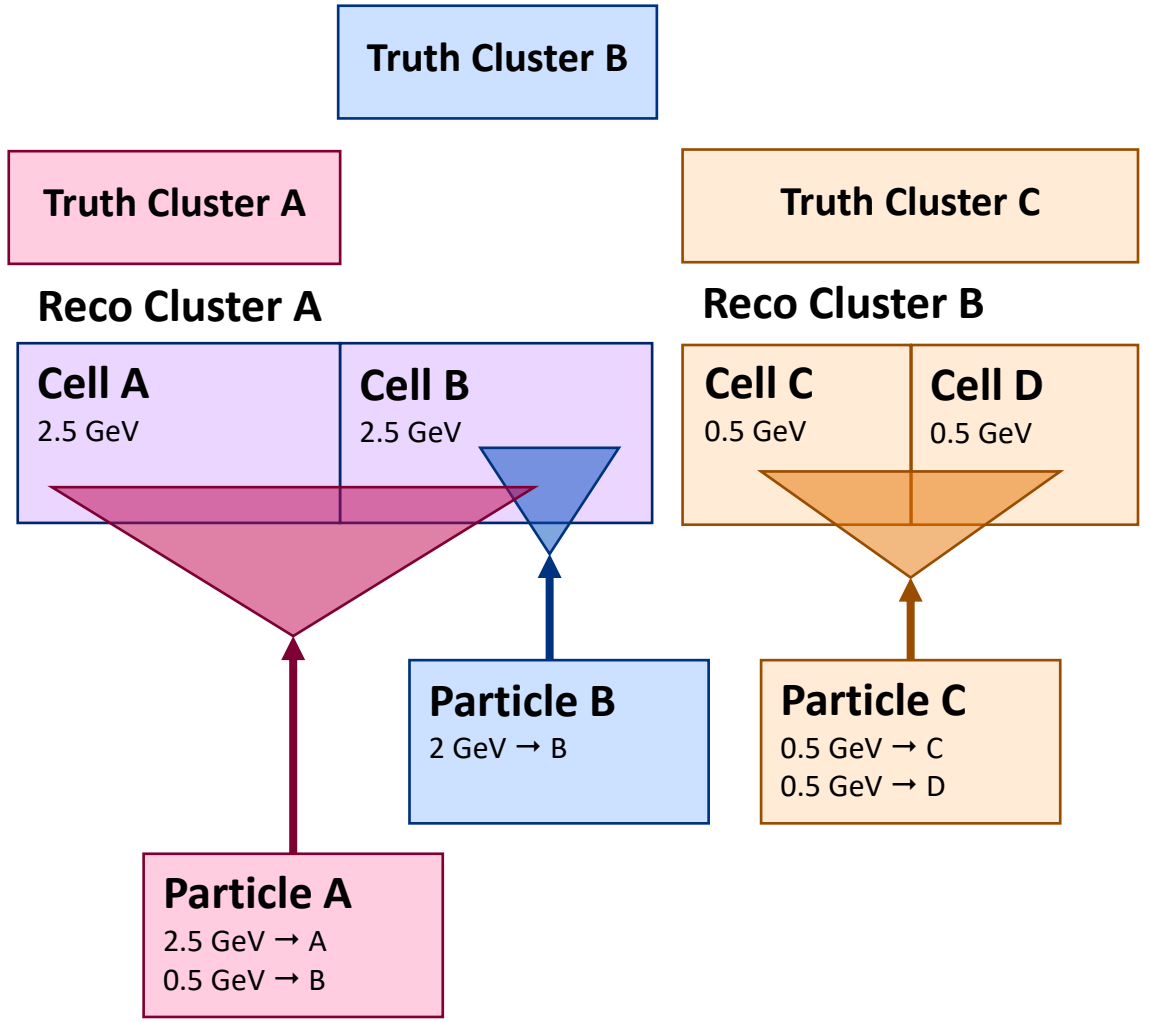
1) Which particles we assign to
   - ☞ For consistency w/ associations, my thought is **primaries (status == 1)**
2) How to assign hits to particles
   - ☞ i.e. what happens when multiple particles contribute to a cluster?



Truth Clustering | Updating Assignment (2/3)

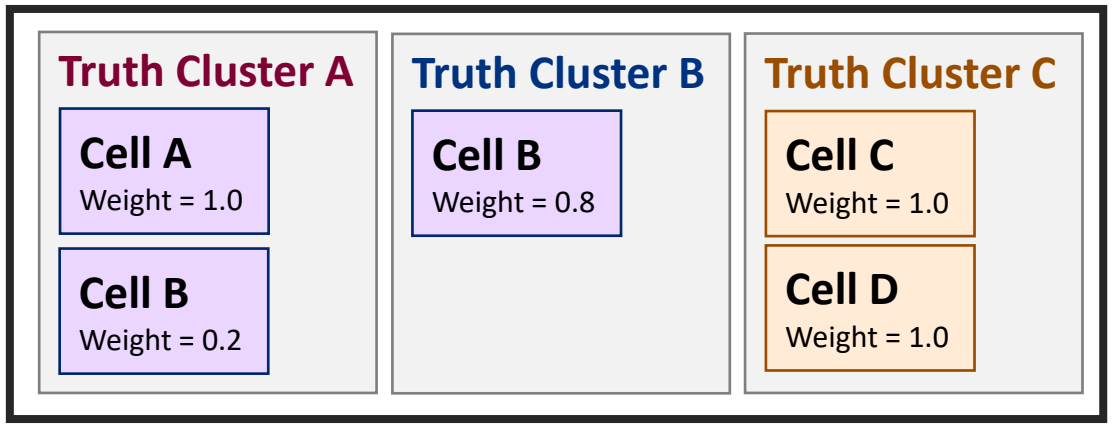
- **Handling multiple contributors:** I see only 2 possible options
  1) **Winner takes all**, particle contributing most gets that hit
  2) **Weighted distribution,** hit gets assigned to all contributors *but* with a weight = fraction contributed by each



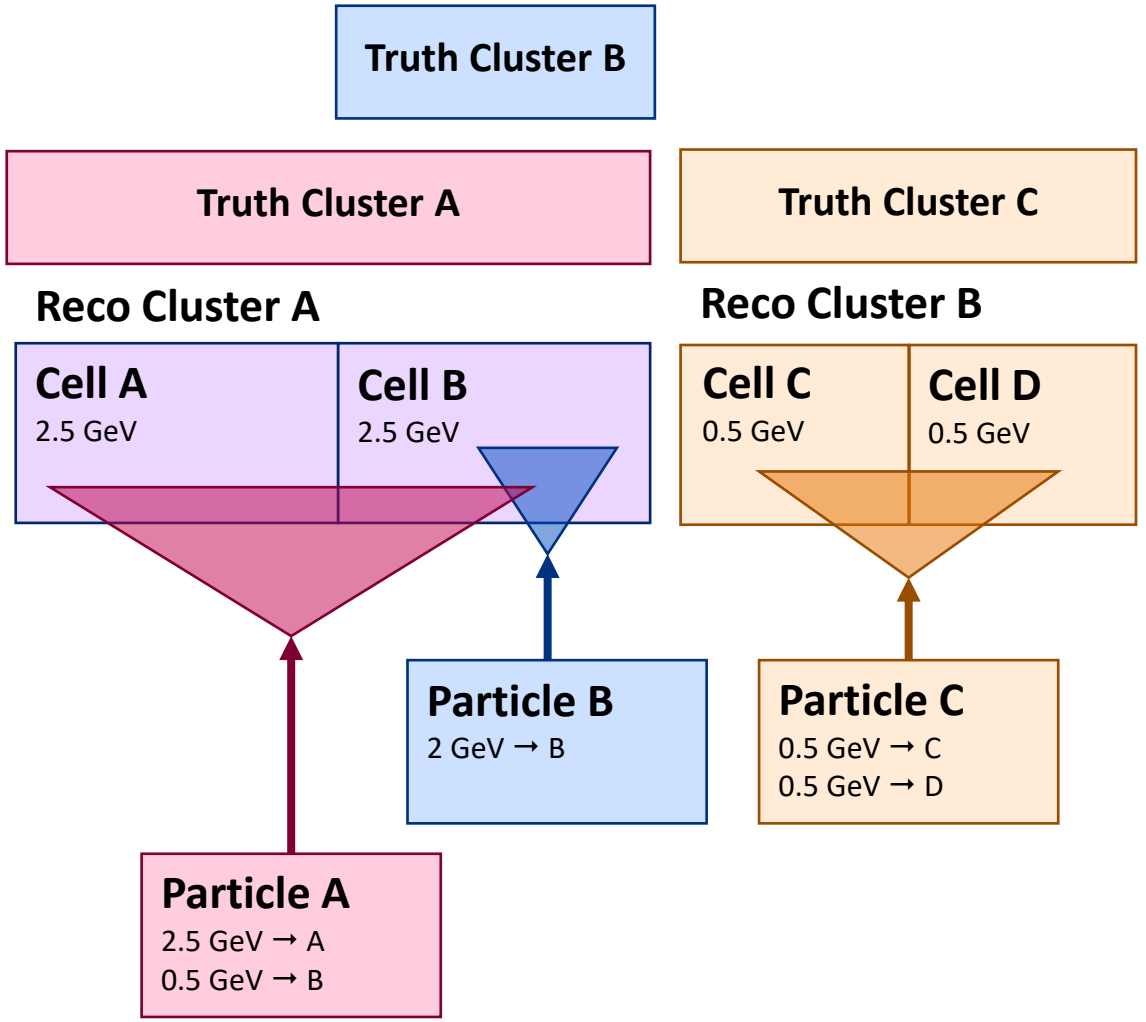


Truth Clustering | Updating Assignment (3/3)

- **Handling multiple contributors:** I see only 2 possible options
  1) **Winner takes all**, particle contributing most gets that hit
  2) **Weighted distribution,** hit gets assigned to all contributors *but* with a weight = fraction contributed by each



Option 2



Discussion | Questions & Notes

- **Two questions to discuss:**
  a) Does assigning to status == 1 particles make sense?
  b) Which assignment method, *Winner Takes All* vs. *Weighted Distribution*, makes sense?
     - ☞ Personally, I lean towards *Weighted Distribution:* seems most aligned w/ our approach to associations…

- **Notes:**
  - Could always make assignment method a configurable option
  - Status == 1 condition should be revisited soon (e.g. see EICrecon#2178), but slightly out of scope for this discussion
  - ☞ **This is a good task for a new student/postdoc!**

Some other possible (but not ideal) paths:

- Remove truth clusters entirely, rely only on MC-Cluster associations
  - › **Not ideal** ∵ truth clusters (can) offer reference to gauge how close our reco clustering algo gets to capturing individual showers

- Group *contributions* into a structure
  - › **Not ideal** ∵ we can't save contributions by default due to space constraints
  - › *And* would require a data-model change