Discussion on SimCalorimeterHitProcessor

Minho Kim
Argonne National Laboratory

BIC Simulation Meeting
May 27, 2025

Argonne NATIONAL LABORATORY

Effect of the CaloHitContribution

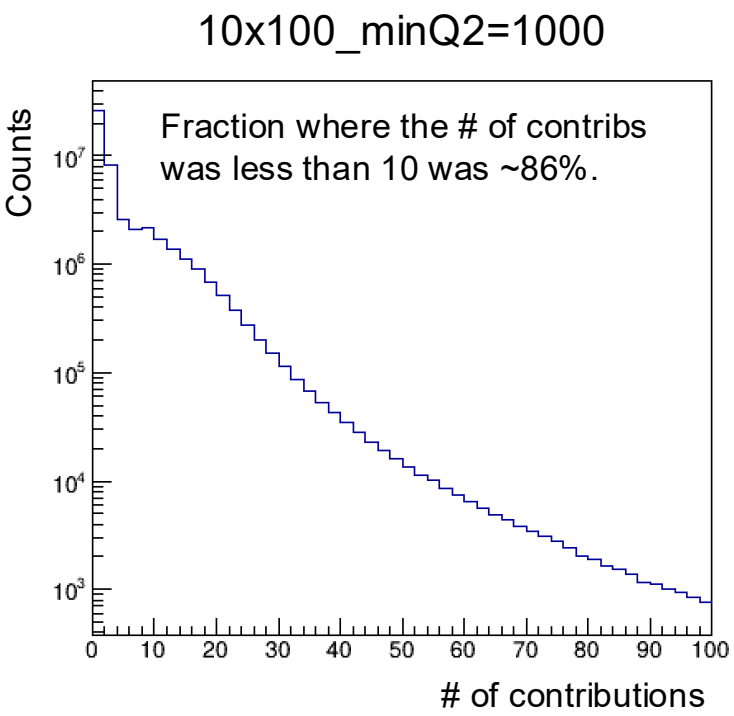
```
// regroup the sim hits by mc particle
for (const auto& ih : *in_hits) {
  for (const auto& contrib : ih.getContributions()) {
    edm4hep::MCParticle primary = get_primary(contrib);

    auto& simhit = mapMCParToSimCalHit[primary].emplace_back(ih.getCellID(), contrib.getEnergy(),
                                                             ih.getPosition());
    simhit.addToContributions(contrib);

    trace("Identified primary: id = {}, pid = {}, total energy = {}, contributed = {}",
          primary.getObjectID().index, primary.getPDG(), primary.getEnergy(),
          mapMCParToSimCalHit[primary].back().getEnergy());
  }
```

wdconinc last week — Member

This grows potentially large. Can you provide some comment on how large you expect this to grow in worst case conditions? Does it affect memory limits on our running of reconstruction?



- To study how large the above for-loop grows, # of hit contributions for each hit was studied using a PYTHIA (10x100_minQ2=1000) sample.
- Fraction where the # of hit contributions was less than 10 was ~86%.
- CalorimterHitDigi, CalorimeterClusterRecoCog, CalorimeterTruthClustering, and ImagingClusterReco are also using the above for-loop. → Using at SimCalorimeterHitProcessor won't cause a bit problem.

Two filling parts

```
    auto out_hit_contrib = out_hit_contribs->create();
    out_hit_contrib.setPDG(leading_contrib.getPDG());
    out_hit_contrib.setEnergy(static_cast<float>(edepSum));
    out_hit_contrib.setTime(timeEar);
    out_hit_contrib.setStepPosition(leading_contrib.getStepPosition());
    out_hit_contrib.setParticle(par);

    auto out_hit = out_hits->create();
    out_hit.setCellID(leading_hit.getCellID());
    out_hit.setEnergy(static_cast<float>(edepSum * attFactor));
    out_hit.setPosition(leading_hit.getPosition());
    out_hit.addToContributions(out_hit_contrib);
  }
} else {
  for (const auto& hit : hits) {
    auto contrib = hit.getContributions(0);

    auto out_hit_contrib = out_hit_contribs->create();
    out_hit_contrib.setPDG(contrib.getPDG());
    out_hit_contrib.setEnergy(contrib.getEnergy());
    out_hit_contrib.setTime(contrib.getTime());
    out_hit_contrib.setStepPosition(contrib.getStepPosition());
    out_hit_contrib.setParticle(par);

    auto out_hit = out_hits->create();
    out_hit.setCellID(hit.getCellID());
    out_hit.setEnergy(hit.getEnergy());
    out_hit.setPosition(hit.getPosition());
    out_hit.addToContributions(out_hit_contrib);
```

When merging hits is necessary, the merged hits are filled here.

When merging hits is not necessary, the original hits that have different counts and elements are filled here.

wdconinc last week

Member

This looks like code duplication that should be avoided.

Converting dd4hep::cm to dd4hep::mm

Because the detector edge position of the BIC is defined by “cm”, it is converted to “mm”.

```
// get reference position for attenuating hits
// convert its unit from cm to mm
if (!m_cfg.attenuationReferencePositionName.empty()) {
  m_attenuationReferencePosition =
      m_geo.detector()->constant<double>(m_cfg.attenuationReferencePositionName) / dd4hep::mm;
}
```

Because the position of the hit has “mm” unit by default, it is passed into the attenuation function as it is.

```
attFactor = get_attenuation(leading_hit.getPosition().z);
```

```
double SimCalorimeterHitProcessor::get_attenuation(double zpos) const {
  double length = std::abs(m_attenuationReferencePosition.value() - zpos);
  double factor = m_cfg.attPars[0] * std::exp(-length / m_cfg.attPars[1]) +
                  (1 - m_cfg.attPars[0]) * std::exp(-length / m_cfg.attPars[2]);
  return factor;
}
```

Parameters of the attenuation function were also defined by “mm”.

```
// Make sure left and right use the same value
decltype(SimCalorimeterHitProcessorConfig::attPars) EcalBarrelScFi_attPars = {
    0.416212, 74.739875 / dd4hep::mm, 752.188383 / dd4hep::mm};
```

Other status

- Basic structure of the CalorimeterPulseGeneration has been completed. Once the SimCalorimeterHitProcessor is merged, it will also be pull-requested.
- Afterwards, PulseCombiner and PulseNoice will also be implemented in order.