MC Recon and Hpstr

- Analysis Workshop 2019 data -

PF, TT, OM, CB

06/04/2020

Introduction / Outline

- The first test passes for the reconstruction of 2019 MC and Data started last week after several updates have been included in the reconstruction code during the past months
- In this talk I'm going to provide
  - Brief summary of **available Data/MC**
    - (I'm not going to talk about checks on data today)
  - A brief summary of changes that went into **track reconstruction code**
  - Available **steering files** for MC reconstruction
    - **LCIO ntuple content** in the produced files
  - **Timing** on current reconstruction pipeline
  - Current **issues** in reconstruction pass
  - Analysis of LCIO ntuples: **hpstr package**
  - **Personal thoughts for discussion on MC production / reconstruction**

Available MC and Data samples

- 2019 Data and MC samples are stored in full at jLab and can be cached / transferred if needed.
  - A certain amount of 2019 data has been processed and stored on SLAC machines. **Be aware of the reconstruction differences and details**:
    - Few partitions for each “good run” of 2019 => old snapshot of hps-java master, v1 detector
    - Large amount of Run 10031 =>
      - Processed with iss687 hps-java branch, v2 detector
    - Skims for 10103 and 9921 for dedicated studies
- The available reconstructed MC made by TT has been processed with branch iss677
- **As the efforts to produce better reconstruction code/LCIO files/calibrations are ongoing, this information will be outdated soon.**

- Useful links about samples location, production details and processing examples are/will be posted at these locations:
  - MC summary page
  - 2019 Data Summary page

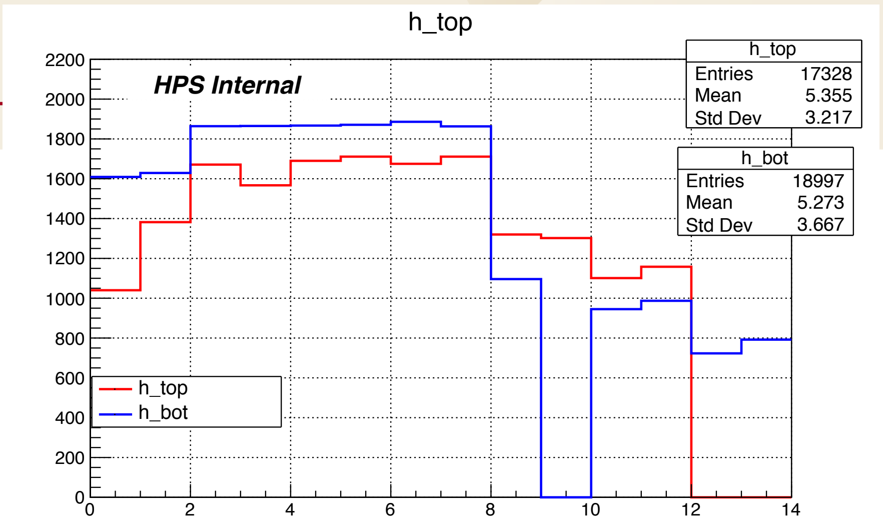
Few different hps-java versions…

- The current hps-java master includes the changes made in iss677:
  - A full review of what went in can be checked in #685 pull request but in few words it includes:
    - **Updated trigger drivers** for MC readout simulation
    - Use of **SiPixel class** for modelling the SVT thin layers
    - Changes to **HelicalTrackHitDriver, GBLRefitter, KalmanPatRecHPS** to run over 2019 geometry with new thin layers
    - Several other minor tweaks/fixes/technicalities..

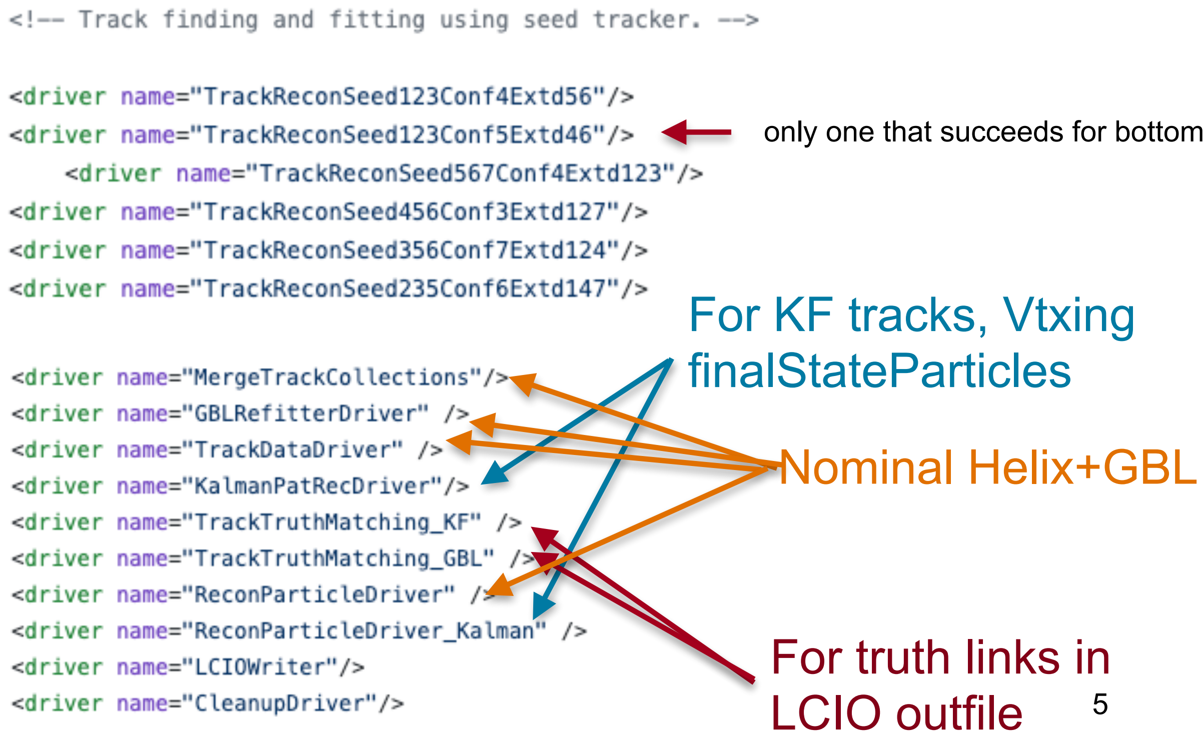
- Together with this branch comes a "recommended" steering-file for checking 2019 MC Reconstruction:
  - PhysicsRun2019MCRecon.lcsim

Reconstruction configuration

- The tri-trig readout sample has been generated with:
  - **Top Ly7 (old ly6) off**
  - **Axial Bottom Ly5 (old ly4) off**
- Quite standard job configuration for Hit formation
- Track Finding uses few strategies: only one succeeds for bottom tracks
- To the nominal reco has been added also KF track finding and fitting interfaced with recon drivers
- TrackTruthMatching is provided for offline studies.
  - Tracks are matched to MCParticles which are used to form TruthTracks for performance checks.



HitsOnTrack for KF Tracks



```
<!-- Track finding and fitting using seed tracker. -->

<driver name="TrackReconSeed123Conf4Extd56"/>
<driver name="TrackReconSeed123Conf5Extd46"/>
    <driver name="TrackReconSeed567Conf4Extd123"/>
<driver name="TrackReconSeed456Conf3Extd127"/>
<driver name="TrackReconSeed356Conf7Extd124"/>
<driver name="TrackReconSeed235Conf6Extd147"/>

<driver name="MergeTrackCollections"/>
<driver name="GBLRefitterDriver" />
<driver name="TrackDataDriver" />
<driver name="KalmanPatRecDriver"/>
<driver name="TrackTruthMatching_KF" />
<driver name="TrackTruthMatching_GBL" />
<driver name="ReconParticleDriver" />
<driver name="ReconParticleDriver_Kalman" />
<driver name="LCIOWriter"/>
<driver name="CleanupDriver"/>
```

only one that succeeds for bottom

For KF tracks, Vtxing finalStateParticles

Nominal Helix+GBL

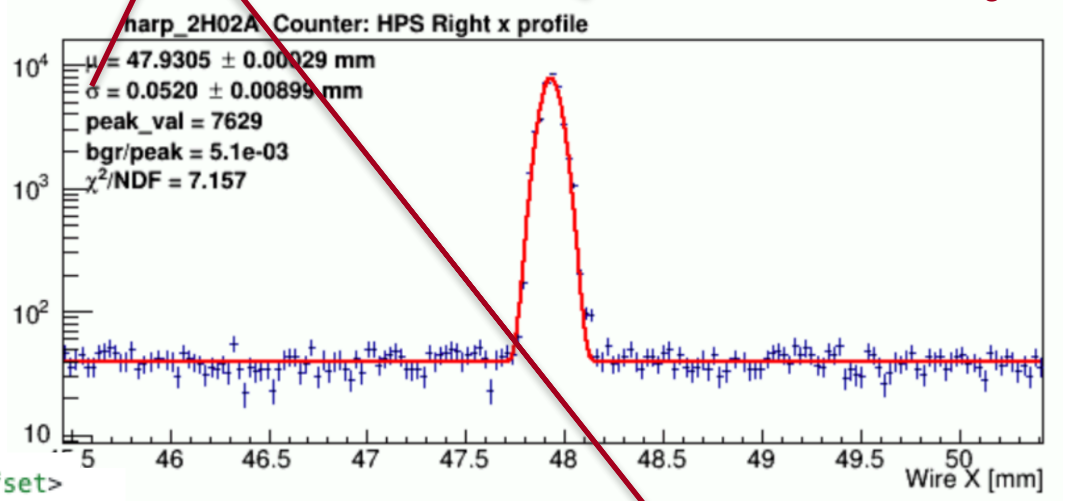
For truth links in LCIO outfile

Reconstruction configuration - ReconParticleDriver

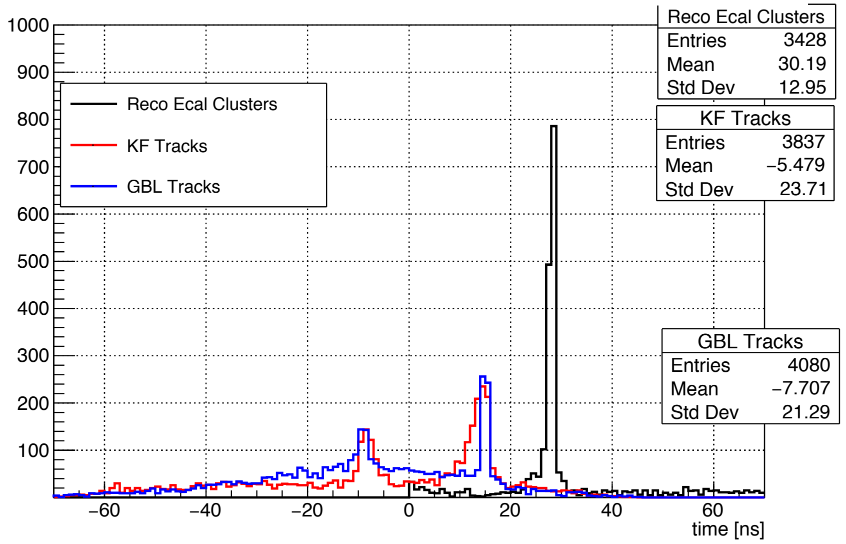
```
<beamPositionX> 0  </beamPositionX>
<beamSigmaX> 0.05 </beamSigmaX>
<beamPositionY> 0 </beamPositionY>
<beamSigmaY> 0.02 </beamSigmaY>
<beamPositionZ> -7.5 </beamPositionZ>
```

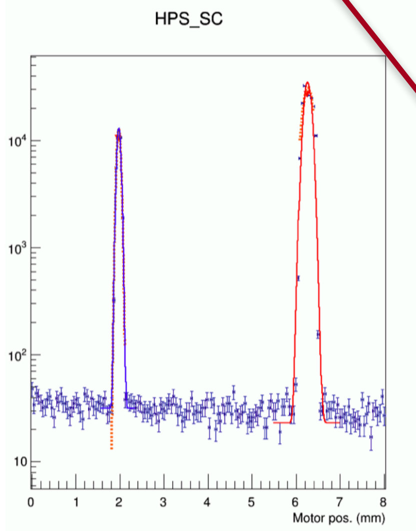
From [MC sim configuration]

- **Vertices** are formed with both Helix+GBL and KF Tracks
  - *Vertices formed with KF Tracks have "_KF" in CollectionName*
- **Vertices are formed without requiring cluster/track matching**
  - *Still working to check track-cluster matching in 2019*
- Nominal settings for BS position (0,0,-7.5)
  Size was taken from SVT wire scan to resemble data (simulation was done with sigma_x(y) = 0 mm)
- TrackClusterTimeOffset from checking ClusterTime distribution
  - Tracks time distribution needs to be cross checked as double peak wasn't expected



```
<trackClusterTimeOffset>28</trackClusterTimeOffset>
```





File: svt_bot_scan_0180.asc

Analyze from HPS_t counter

Fit full peak
bot_mot_pos1 = 1.976 mm
bot_mot_pos2 = 6.258 mm
bot_wire_dist = 1.990 mm
bot_beam_Y = -0.155 ± 0.000092 mm
bot_beam_X = 0.257 ± 0.000764 mm
bot_beam_σ_Y = 0.0193 ± 0.069238 mm

Fit peak ± 3 bins
bot_mot_pos1 = 1.976 mm
bot_mot_pos2 = 6.270 mm
bot_wire_dist = 1.996 mm
bot_beam_Y = -0.155 ± 0.000092 mm
bot_beam_X = 0.294 ± 0.000588 mm
bot_beam_σ_Y = 0.0193 ± 0.068872 mm

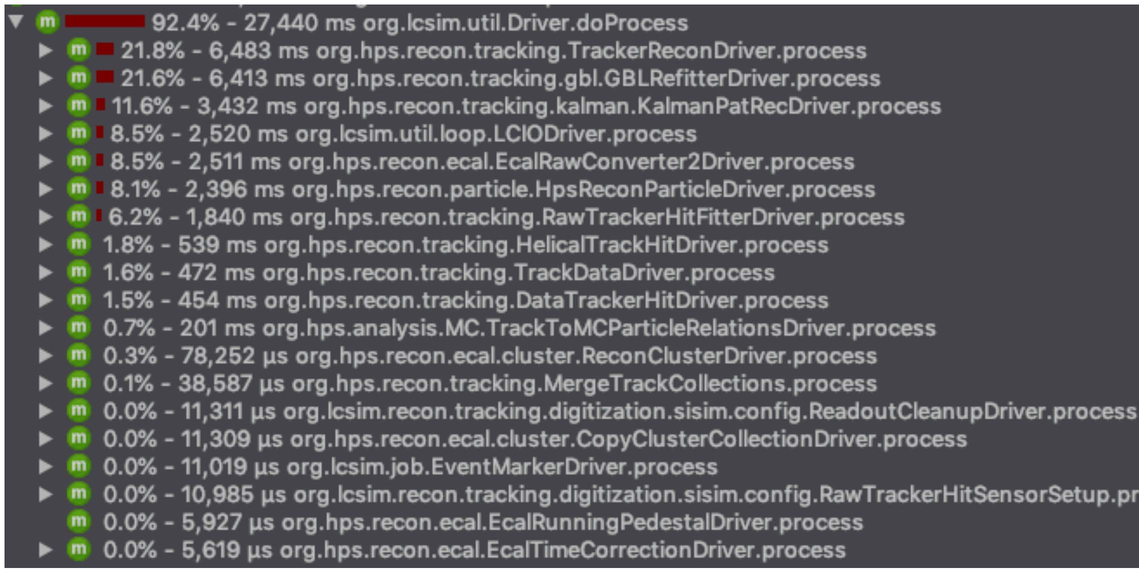
Analyze from HPS_SC counter

Fit full peak
bot_mot_pos1 = 1.977 mm
bot_mot_pos2 = 6.257 mm
bot_wire_dist = 1.989 mm
bot_beam_Y = -0.154 ± 0.000128 mm
bot_beam_X = 0.252 ± 0.001054 mm
bot_beam_σ_Y = 0.0199 ± 0.064258 mm

Fit peak ± 3 bins
bot_mot_pos1 = 1.977 mm
bot_mot_pos2 = 6.269 mm
bot_wire_dist = 1.995 mm
bot_beam_Y = -0.154 ± 0.000104 mm
bot_beam_X = 0.289 ± 0.000805 mm
bot_beam_σ_Y = 0.0202 ± 0.142527 mm

Processing time - Tri-Trig ***without Beam***

- A summary breakdown of the CPU time spent in MC processing is shown
- File tested: /nfs/slac/g/hps3/mc/mc_2019/readout/tritrig/singles/4pt5/tritrig_123.slcio
- With the current strategies, tracking takes:
  - ~22% in seeding and global fitting stage
  - ~22% in GBL Refitting stage
- Kalman track finding and fitting takes ~12% of the event time
- Some non-negligible amount of time is spent in the HpsReconParticleDriver (8%) and RawHit Fitting (6%)



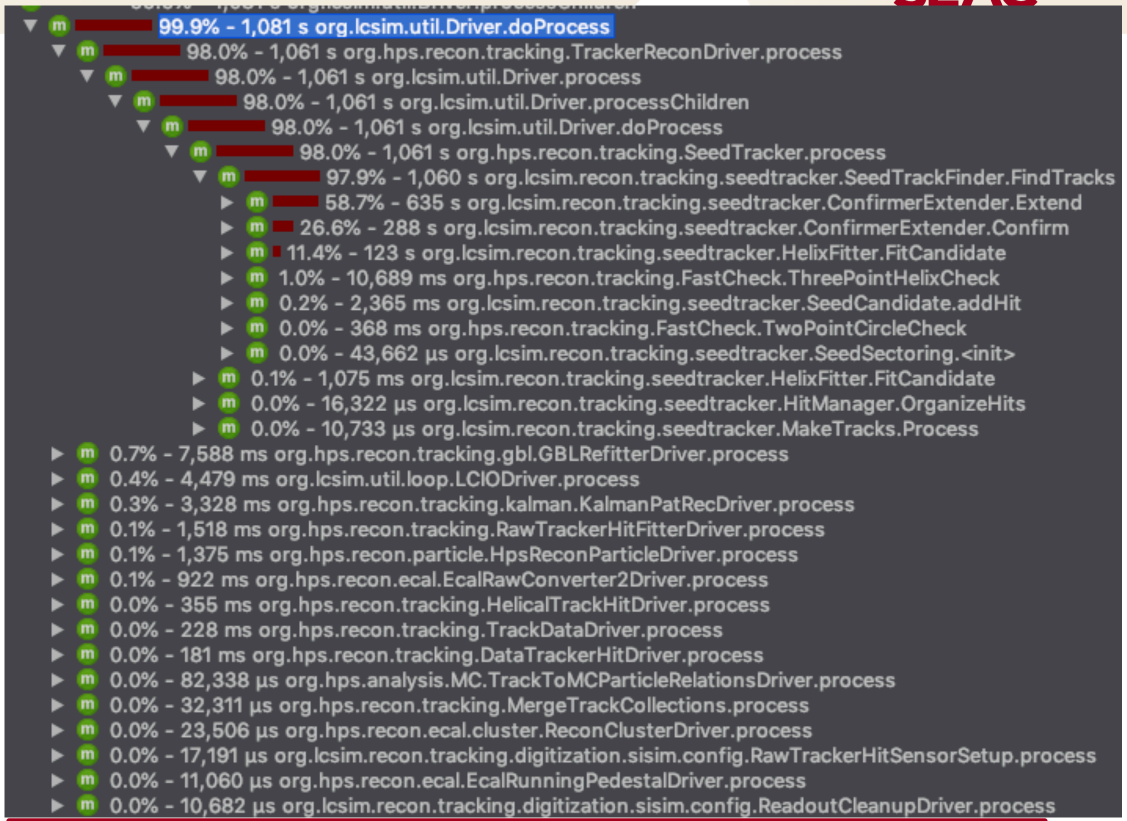
```
▼ m 92.4% - 27,440 ms org.lcsim.util.Driver.doProcess
  ▶ m 21.8% - 6,483 ms org.hps.recon.tracking.TrackerReconDriver.process
  ▶ m 21.6% - 6,413 ms org.hps.recon.tracking.gbl.GBLRefitterDriver.process
  ▶ m 11.6% - 3,432 ms org.hps.recon.tracking.kalman.KalmanPatRecDriver.process
  ▶ m 8.5% - 2,520 ms org.lcsim.util.loop.LCIODriver.process
  ▶ m 8.5% - 2,511 ms org.hps.recon.ecal.EcalRawConverter2Driver.process
  ▶ m 8.1% - 2,396 ms org.hps.recon.particle.HpsReconParticleDriver.process
  ▶ m 6.2% - 1,840 ms org.hps.recon.tracking.RawTrackerHitFitterDriver.process
  ▶ m 1.8% - 539 ms org.hps.recon.tracking.HelicalTrackHitDriver.process
  ▶ m 1.6% - 472 ms org.hps.recon.tracking.TrackDataDriver.process
  ▶ m 1.5% - 454 ms org.hps.recon.tracking.DataTrackerHitDriver.process
  ▶ m 0.7% - 201 ms org.hps.analysis.MC.TrackToMCParticleRelationsDriver.process
  ▶ m 0.3% - 78,252 µs org.hps.recon.ecal.cluster.ReconClusterDriver.process
  ▶ m 0.1% - 38,587 µs org.hps.recon.tracking.MergeTrackCollections.process
  ▶ m 0.0% - 11,311 µs org.lcsim.recon.tracking.digitization.sisim.config.ReadoutCleanupDriver.process
  ▶ m 0.0% - 11,309 µs org.hps.recon.ecal.cluster.CopyClusterCollectionDriver.process
  ▶ m 0.0% - 11,019 µs org.lcsim.job.EventMarkerDriver.process
  ▶ m 0.0% - 10,985 µs org.lcsim.recon.tracking.digitization.sisim.config.RawTrackerHitSensorSetup.pr
    m 0.0% - 5,927 µs org.hps.recon.ecal.EcalRunningPedestalDriver.process
  ▶ m 0.0% - 5,619 µs org.hps.recon.ecal.EcalTimeCorrectionDriver.process
```

Total Tracking time ~40% in tri-trig signal without beam background
See Backup for a more detailed dump

jProfiler
Evaluation version, remotely attached to cent7a, readout to LCIO step

Processing time - Tri-Trig ***with Beam***

- Drastic increase in tracking time
- File tested: /nfs/slac/g/hps3/users/bravo/mc/mc2019/tritrig/readoutFromJLAb/tritrig_1.slcio
- With the current strategies, tracking takes:
  - **~98% in the SeedTracker** (60% in the extension, 27% in the confirmation, 12% in the fitting)
  - Mostly due to very large cuts in rmsTime in SeedTracker (1000ns), but also setting it at (20ns) doesn't help (98% => 93% see Backup)
  - ~0.7% in GBL Refitting stage
- Kalman track finding and fitting takes ~0.3% of the event time in this conditions
- All the rest of the event reconstruction time becomes negligible
- **Not sustainable for high-stat MC or reReco passes.**
- **Total time: 25m for ~150 events on cent7a => 10s/event**



```
▼ 99.9% - 1,081 s org.lcsim.util.Driver.doProcess
  ▼ 98.0% - 1,061 s org.hps.recon.tracking.TrackerReconDriver.process
    ▼ 98.0% - 1,061 s org.lcsim.util.Driver.process
      ▼ 98.0% - 1,061 s org.lcsim.util.Driver.processChildren
        ▼ 98.0% - 1,061 s org.lcsim.util.Driver.doProcess
          ▼ 98.0% - 1,061 s org.hps.recon.tracking.SeedTracker.process
            ▼ 97.9% - 1,060 s org.lcsim.recon.tracking.seedtracker.SeedTrackFinder.FindTracks
              ▶ 58.7% - 635 s org.lcsim.recon.tracking.seedtracker.ConfirmerExtender.Extend
              ▶ 26.6% - 288 s org.lcsim.recon.tracking.seedtracker.ConfirmerExtender.Confirm
              ▶ 11.4% - 123 s org.lcsim.recon.tracking.seedtracker.HelixFitter.FitCandidate
              ▶ 1.0% - 10,689 ms org.hps.recon.tracking.FastCheck.ThreePointHelixCheck
              ▶ 0.2% - 2,365 ms org.lcsim.recon.tracking.seedtracker.SeedCandidate.addHit
              ▶ 0.0% - 368 ms org.hps.recon.tracking.FastCheck.TwoPointCircleCheck
              ▶ 0.0% - 43,662 µs org.lcsim.recon.tracking.seedtracker.SeedSectoring.<init>
            ▶ 0.1% - 1,075 ms org.lcsim.recon.tracking.seedtracker.HelixFitter.FitCandidate
            ▶ 0.0% - 16,322 µs org.lcsim.recon.tracking.seedtracker.HitManager.OrganizeHits
            ▶ 0.0% - 10,733 µs org.lcsim.recon.tracking.seedtracker.MakeTracks.Process
  ▶ 0.7% - 7,588 ms org.hps.recon.tracking.gbl.GBLRefitterDriver.process
  ▶ 0.4% - 4,479 ms org.lcsim.util.loop.LCIODriver.process
  ▶ 0.3% - 3,328 ms org.hps.recon.tracking.kalman.KalmanPatRecDriver.process
  ▶ 0.1% - 1,518 ms org.hps.recon.tracking.RawTrackerHitFitterDriver.process
  ▶ 0.1% - 1,375 ms org.hps.recon.particle.HpsReconParticleDriver.process
  ▶ 0.1% - 922 ms org.hps.recon.ecal.EcalRawConverter2Driver.process
  ▶ 0.0% - 355 ms org.hps.recon.tracking.HelicalTrackHitDriver.process
  ▶ 0.0% - 228 ms org.hps.recon.tracking.TrackDataDriver.process
  ▶ 0.0% - 181 ms org.hps.recon.tracking.DataTrackerHitDriver.process
  ▶ 0.0% - 82,338 µs org.hps.analysis.MC.TrackToMCParticleRelationsDriver.process
  ▶ 0.0% - 32,311 µs org.hps.recon.tracking.MergeTrackCollections.process
  ▶ 0.0% - 23,506 µs org.hps.recon.ecal.cluster.ReconClusterDriver.process
  ▶ 0.0% - 17,191 µs org.lcsim.recon.tracking.digitization.sisim.config.RawTrackerHitSensorSetup.process
  ▶ 0.0% - 11,060 µs org.hps.recon.ecal.EcalRunningPedestalDriver.process
  ▶ 0.0% - 10,682 µs org.lcsim.recon.tracking.digitization.sisim.config.ReadoutCleanupDriver.process
```

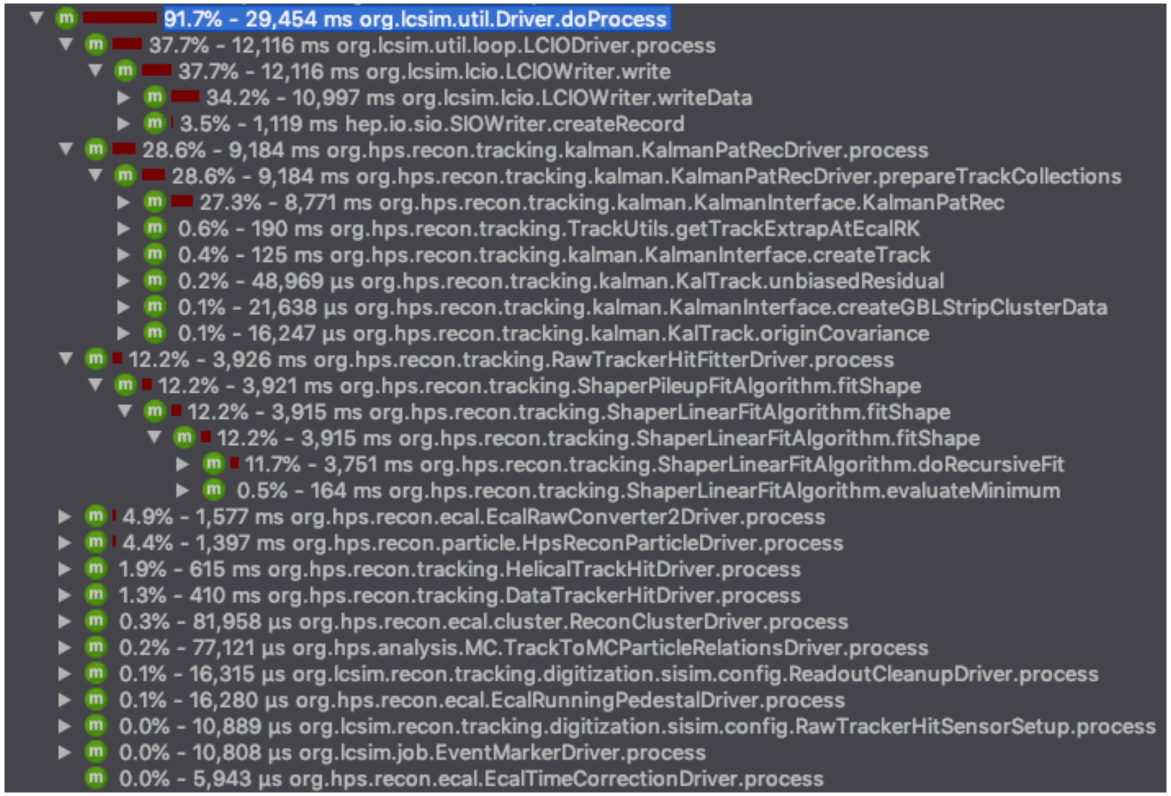
Total Tracking time ~98% in tri-trig signal with beam background. Not sustainable in long run. A more detailed dump in the backup

jProfiler

Evaluation version, remotely attached to cent7a, readout to LCIO step

Processing time - Tri-Trig ***with Beam*** KF Only

- Tested Kalman only reconstruction
- Kalman track finding and fitting takes ~30% of the event time in this conditions
- Writing LCIO output takes **~40%**
- Something can be recovered from SvtRawHitFitting and HPSReconDrivers
- Only ideal => GBL refitter should run on KF Tracks.
- **Total time: 1m40s for ~860 events on cent7a => 0.11s/ event**



```
▼ m 91.7% - 29,454 ms org.lcsim.util.Driver.doProcess
  ▼ m 37.7% - 12,116 ms org.lcsim.util.loop.LCIODriver.process
    ▼ m 37.7% - 12,116 ms org.lcsim.lcio.LCIOWriter.write
      ▶ m 34.2% - 10,997 ms org.lcsim.lcio.LCIOWriter.writeData
      ▶ m 3.5% - 1,119 ms hep.io.sio.SIOWriter.createRecord
  ▼ m 28.6% - 9,184 ms org.hps.recon.tracking.kalman.KalmanPatRecDriver.process
    ▼ m 28.6% - 9,184 ms org.hps.recon.tracking.kalman.KalmanPatRecDriver.prepareTrackCollections
      ▶ m 27.3% - 8,771 ms org.hps.recon.tracking.kalman.KalmanInterface.KalmanPatRec
      ▶ m 0.6% - 190 ms org.hps.recon.tracking.TrackUtils.getTrackExtrapAtEcalRK
      ▶ m 0.4% - 125 ms org.hps.recon.tracking.kalman.KalmanInterface.createTrack
      ▶ m 0.2% - 48,969 µs org.hps.recon.tracking.kalman.KalTrack.unbiasedResidual
      ▶ m 0.1% - 21,638 µs org.hps.recon.tracking.kalman.KalmanInterface.createGBLStripClusterData
      ▶ m 0.1% - 16,247 µs org.hps.recon.tracking.kalman.KalTrack.originCovariance
  ▼ m 12.2% - 3,926 ms org.hps.recon.tracking.RawTrackerHitFitterDriver.process
    ▼ m 12.2% - 3,921 ms org.hps.recon.tracking.ShaperPileupFitAlgorithm.fitShape
      ▼ m 12.2% - 3,915 ms org.hps.recon.tracking.ShaperLinearFitAlgorithm.fitShape
        ▼ m 12.2% - 3,915 ms org.hps.recon.tracking.ShaperLinearFitAlgorithm.fitShape
          ▶ m 11.7% - 3,751 ms org.hps.recon.tracking.ShaperLinearFitAlgorithm.doRecursiveFit
          ▶ m 0.5% - 164 ms org.hps.recon.tracking.ShaperLinearFitAlgorithm.evaluateMinimum
  ▶ m 4.9% - 1,577 ms org.hps.recon.ecal.EcalRawConverter2Driver.process
  ▶ m 4.4% - 1,397 ms org.hps.recon.particle.HpsReconParticleDriver.process
  ▶ m 1.9% - 615 ms org.hps.recon.tracking.HelicalTrackHitDriver.process
  ▶ m 1.3% - 410 ms org.hps.recon.tracking.DataTrackerHitDriver.process
  ▶ m 0.3% - 81,958 µs org.hps.recon.ecal.cluster.ReconClusterDriver.process
  ▶ m 0.2% - 77,121 µs org.hps.analysis.MC.TrackToMCParticleRelationsDriver.process
  ▶ m 0.1% - 16,315 µs org.lcsim.recon.tracking.digitization.sisim.config.ReadoutCleanupDriver.process
  ▶ m 0.1% - 16,280 µs org.hps.recon.ecal.EcalRunningPedestalDriver.process
  ▶ m 0.0% - 10,889 µs org.lcsim.recon.tracking.digitization.sisim.config.RawTrackerHitSensorSetup.process
  ▶ m 0.0% - 10,808 µs org.lcsim.job.EventMarkerDriver.process
    m 0.0% - 5,943 µs org.hps.recon.ecal.EcalTimeCorrectionDriver.process
```

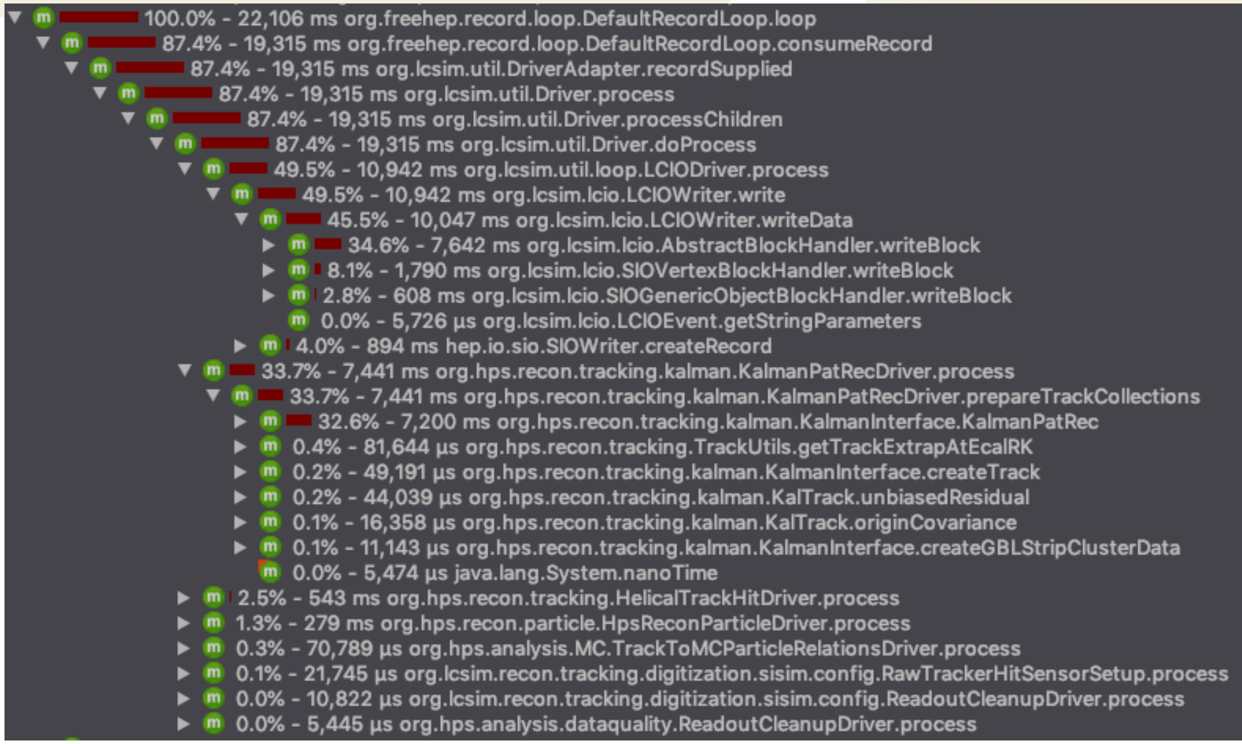
Writing output data is slower than KF tracking, second slowest.
Hit Fitting is a considerable time. Vtxing ~5%

jProfiler

Evaluation version, remotely attached to cent7a, readout to LCIO step

Re-reco from LCIO steering files

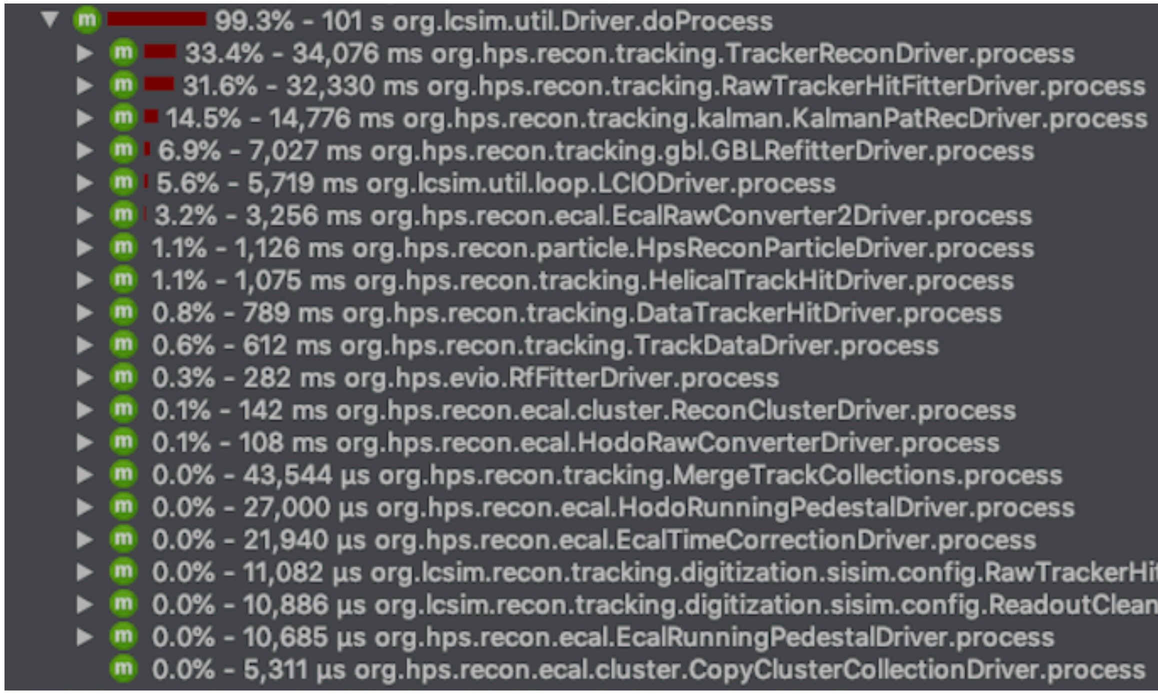
- Some time can be saved if running from pre-reconstructed LCIO files, cleaning up proper containers
- I've made a steering file to run on MC from pre-reconstructed LCIO files: iss687_dev => PhysicsRun2019MCRecon_LCIO.lcsim
- Save 12% processing time from RawFitting
- If ran with KalmanOnly: **~0.09s / evt**



```
▼ 100.0% - 22,106 ms org.freehep.record.loop.DefaultRecordLoop.loop
  ▼ 87.4% - 19,315 ms org.freehep.record.loop.DefaultRecordLoop.consumeRecord
    ▼ 87.4% - 19,315 ms org.lcsim.util.DriverAdapter.recordSupplied
      ▼ 87.4% - 19,315 ms org.lcsim.util.Driver.process
        ▼ 87.4% - 19,315 ms org.lcsim.util.Driver.processChildren
          ▼ 87.4% - 19,315 ms org.lcsim.util.Driver.doProcess
            ▼ 49.5% - 10,942 ms org.lcsim.util.loop.LCIODriver.process
              ▼ 49.5% - 10,942 ms org.lcsim.lcio.LCIOWriter.write
                ▼ 45.5% - 10,047 ms org.lcsim.lcio.LCIOWriter.writeData
                  ▶ 34.6% - 7,642 ms org.lcsim.lcio.AbstractBlockHandler.writeBlock
                  ▶ 8.1% - 1,790 ms org.lcsim.lcio.SIOVertexBlockHandler.writeBlock
                  ▶ 2.8% - 608 ms org.lcsim.lcio.SIOGenericObjectBlockHandler.writeBlock
                    0.0% - 5,726 µs org.lcsim.lcio.LCIOEvent.getStringParameters
                ▶ 4.0% - 894 ms hep.io.sio.SIOWriter.createRecord
            ▼ 33.7% - 7,441 ms org.hps.recon.tracking.kalman.KalmanPatRecDriver.process
              ▼ 33.7% - 7,441 ms org.hps.recon.tracking.kalman.KalmanPatRecDriver.prepareTrackCollections
                ▶ 32.6% - 7,200 ms org.hps.recon.tracking.kalman.KalmanInterface.KalmanPatRec
                ▶ 0.4% - 81,644 µs org.hps.recon.tracking.TrackUtils.getTrackExtrapAtEcalRK
                ▶ 0.2% - 49,191 µs org.hps.recon.tracking.kalman.KalmanInterface.createTrack
                ▶ 0.2% - 44,039 µs org.hps.recon.tracking.kalman.KalTrack.unbiasedResidual
                ▶ 0.1% - 16,358 µs org.hps.recon.tracking.kalman.KalTrack.originCovariance
                ▶ 0.1% - 11,143 µs org.hps.recon.tracking.kalman.KalmanInterface.createGBLStripClusterData
                  0.0% - 5,474 µs java.lang.System.nanoTime
            ▶ 2.5% - 543 ms org.hps.recon.tracking.HelicalTrackHitDriver.process
            ▶ 1.3% - 279 ms org.hps.recon.particle.HpsReconParticleDriver.process
            ▶ 0.3% - 70,789 µs org.hps.analysis.MC.TrackToMCParticleRelationsDriver.process
            ▶ 0.1% - 21,745 µs org.lcsim.recon.tracking.digitization.sisim.config.RawTrackerHitSensorSetup.process
            ▶ 0.0% - 10,822 µs org.lcsim.recon.tracking.digitization.sisim.config.ReadoutCleanupDriver.process
            ▶ 0.0% - 5,445 µs org.hps.analysis.dataquality.ReadoutCleanupDriver.process
```

Bonus: Checks on 2019 Data Run 10031

- Checked Reco Time on Data
- File tested: /nfs/slac/g/hps_data2/data/physrun2019/hps_010031/hps_010031.evio.00054
- **SeedTracker and HelixFitting take ~33%**
- **RawHitFitting takes 32%** of processing time, partly due to monster events rate.
- **KF up to 15%, GBL Refitting 7%**
- **Writing data about 5% of the time**
- **50 seconds for 400 events from evio->LCIO: 0.125s / event**



```
▼ m 99.3% - 101 s org.lcsim.util.Driver.doProcess
  ▶ m 33.4% - 34,076 ms org.hps.recon.tracking.TrackerReconDriver.process
  ▶ m 31.6% - 32,330 ms org.hps.recon.tracking.RawTrackerHitFitterDriver.process
  ▶ m 14.5% - 14,776 ms org.hps.recon.tracking.kalman.KalmanPatRecDriver.process
  ▶ m 6.9% - 7,027 ms org.hps.recon.tracking.gbl.GBLRefitterDriver.process
  ▶ m 5.6% - 5,719 ms org.lcsim.util.loop.LCIODriver.process
  ▶ m 3.2% - 3,256 ms org.hps.recon.ecal.EcalRawConverter2Driver.process
  ▶ m 1.1% - 1,126 ms org.hps.recon.particle.HpsReconParticleDriver.process
  ▶ m 1.1% - 1,075 ms org.hps.recon.tracking.HelicalTrackHitDriver.process
  ▶ m 0.8% - 789 ms org.hps.recon.tracking.DataTrackerHitDriver.process
  ▶ m 0.6% - 612 ms org.hps.recon.tracking.TrackDataDriver.process
  ▶ m 0.3% - 282 ms org.hps.evio.RfFitterDriver.process
  ▶ m 0.1% - 142 ms org.hps.recon.ecal.cluster.ReconClusterDriver.process
  ▶ m 0.1% - 108 ms org.hps.recon.ecal.HodoRawConverterDriver.process
  ▶ m 0.0% - 43,544 µs org.hps.recon.tracking.MergeTrackCollections.process
  ▶ m 0.0% - 27,000 µs org.hps.recon.ecal.HodoRunningPedestalDriver.process
  ▶ m 0.0% - 21,940 µs org.hps.recon.ecal.EcalTimeCorrectionDriver.process
  ▶ m 0.0% - 11,082 µs org.lcsim.recon.tracking.digitization.sisim.config.RawTrackerHit
  ▶ m 0.0% - 10,886 µs org.lcsim.recon.tracking.digitization.sisim.config.ReadoutClean
  ▶ m 0.0% - 10,685 µs org.hps.recon.ecal.EcalRunningPedestalDriver.process
    m 0.0% - 5,311 µs org.hps.recon.ecal.cluster.CopyClusterCollectionDriver.process
```

Data processing will be slow with current processing strategy. Something can be

jProfiler

Evaluation version, remotely attached to cent7a, evil to LCIO step

Few words on timing checks

- I’ve tried to time the various reconstruction configurations to have a feeling of how long will take to process and calibrate 2019 data
- I used jProfiler. It’s the first time I use such profiler, so some things might not be very precise.
- Disclaimer:

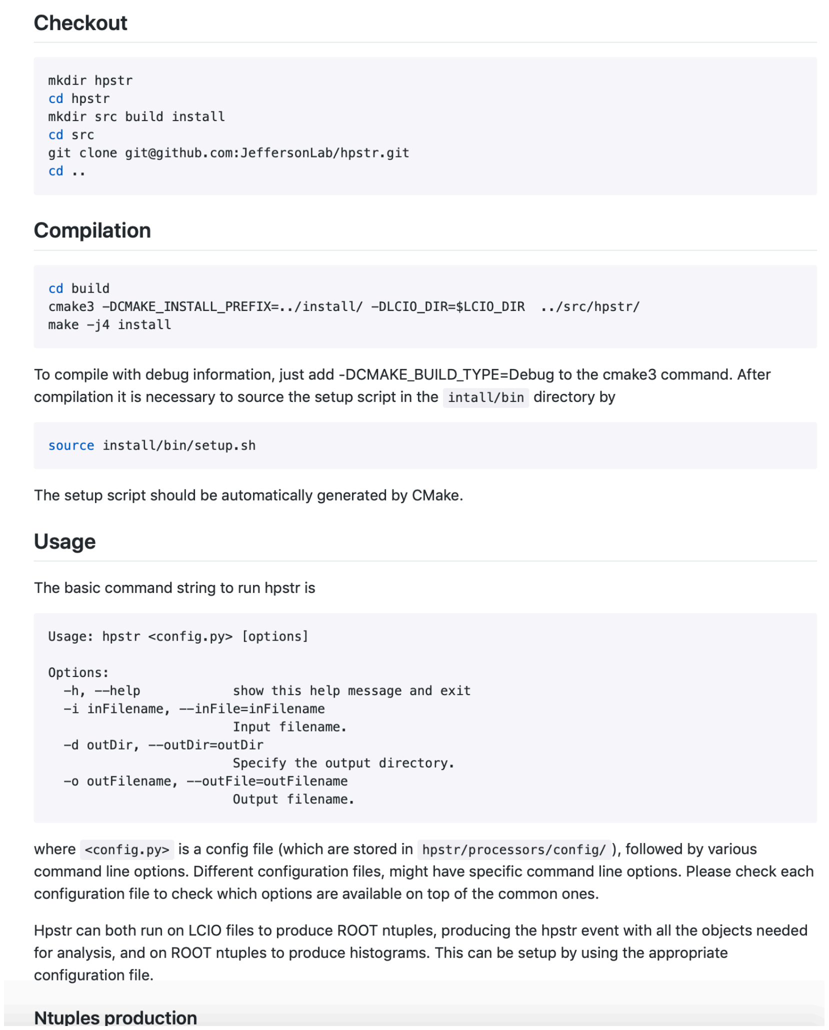
  - MC without beam can be compared with MC with beam as the same steering file was used.

  - Data cannot be directly compared to MC processing as some extra cleaning of the events is applied.
    - SeedTracker rmsTimeCut 1000ns (20ns) in MC (Data)
    - maxTrackerHits 250 (200) in MC (Data)
- **However the checks still give a feeling of the processing time with the current SW and steering files.**

Introduction

- Hpstr (Heavy Photon Search Toolkit for Reconstruction) is a C++/python based package for data analysis.
- The package does:
  - Conversion from **LCIO -> ROOT ntuples** (a la DST), defining a ROOT based EDM with **objects**, i.e. tracks/particles/vertices, and **links** (TRef) between them
  - It provides ROOT tuples processing to produce **histograms** and/or **flat-Ntuples** (even if in principle one can do the same from LCIO if needed)
  - **Provides a** Post processing of **histograms** or **flat-Ntuples** as well.

- The package repository:
  - https://github.com/JeffersonLab/hpstr
  - A README is provided with full instructions from checkout to processing LCIO files
- **A full description of the actual content and structure of the package is beyond the scope of this talk today**



Checkout

```
mkdir hpstr
cd hpstr
mkdir src build install
cd src
git clone git@github.com:JeffersonLab/hpstr.git
cd ..
```

Compilation

```
cd build
cmake3 -DCMAKE_INSTALL_PREFIX=../install/ -DLCIO_DIR=$LCIO_DIR  ../src/hpstr/
make -j4 install
```

To compile with debug information, just add -DCMAKE_BUILD_TYPE=Debug to the cmake3 command. After compilation it is necessary to source the setup script in the `intall/bin` directory by

```
source install/bin/setup.sh
```

The setup script should be automatically generated by CMake.

Usage

The basic command string to run hpstr is

```
Usage: hpstr <config.py> [options]

Options:
  -h, --help            show this help message and exit
  -i inFilename, --inFile=inFilename
                        Input filename.
  -d outDir, --outDir=outDir
                        Specify the output directory.
  -o outFilename, --outFile=outFilename
                        Output filename.
```

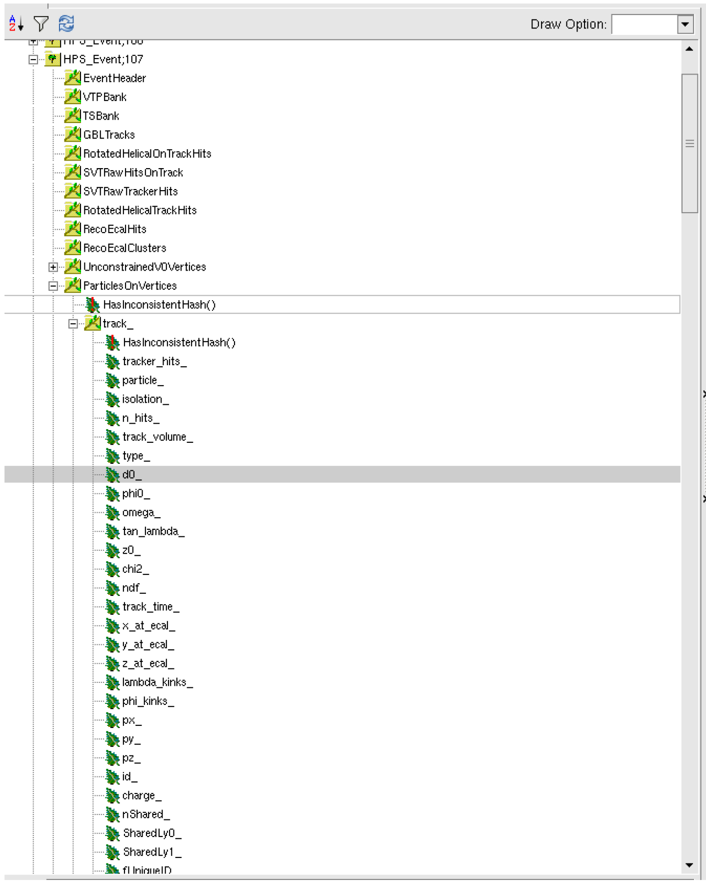
where `<config.py>` is a config file (which are stored in `hpstr/processors/config/`), followed by various command line options. Different configuration files, might have specific command line options. Please check each configuration file to check which options are available on top of the common ones.

Hpstr can both run on LCIO files to produce ROOT ntuples, producing the hpstr event with all the objects needed for analysis, and on ROOT ntuples to produce histograms. This can be setup by using the appropriate configuration file.

Ntuples production

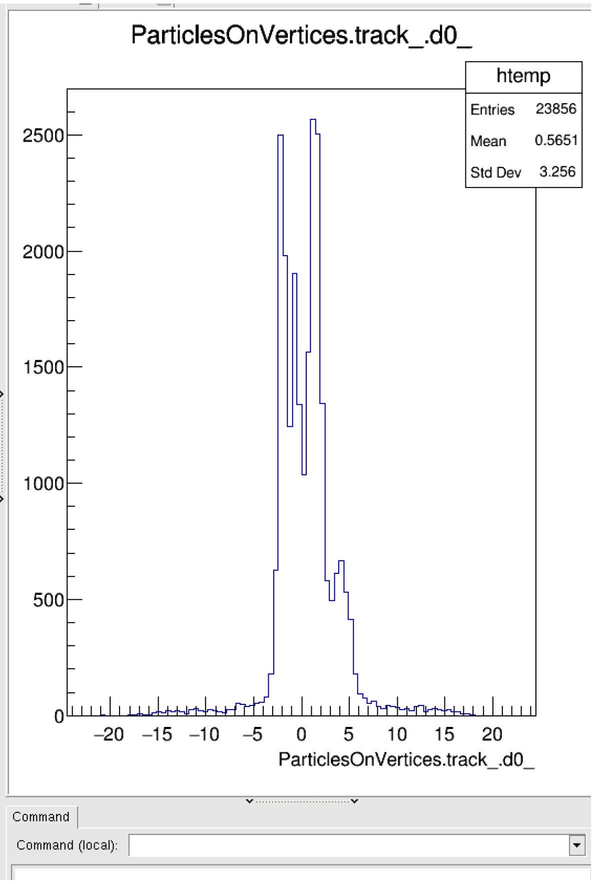
Current Status

- HPSTR has now a fully implemented processing of LCIO to a ROOT n-tuple that has the structure for performing both **vertex** and **BH** analysis
- Content:
  - **Unconstrained and TargetConstrained V0s**
  - **Particles** containing associated tracks and clusters
  - **All tracks** of the events containing the **hits on tracks**
  - **All clusters** in the Calo containing **Calo hits**
  - **Event Information including trigger flags**
- Apart from that, single processors can be defined for specific tasks



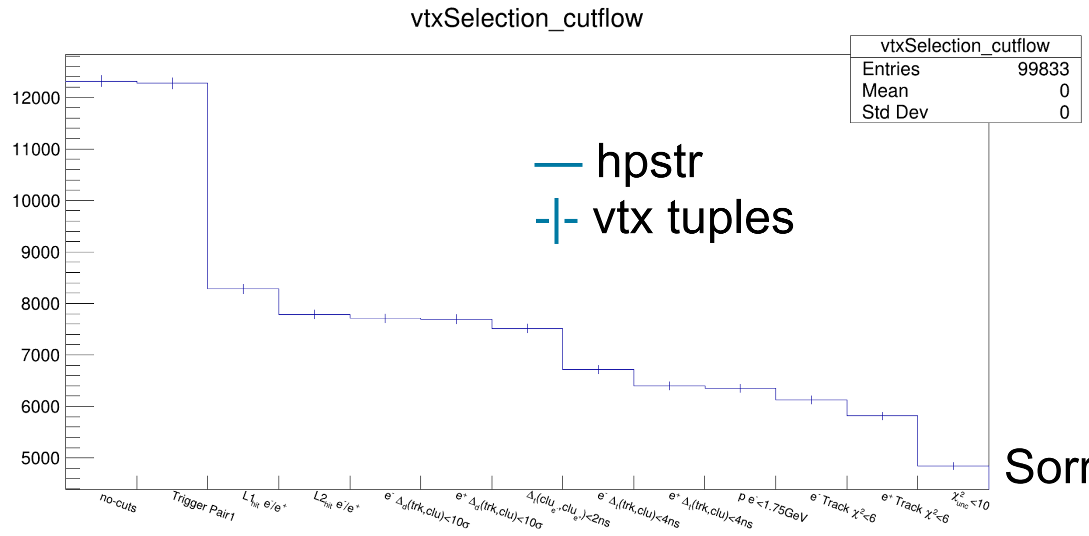
Draw Option:

```
HPS_Event;107
  EventHeader
  VTPBank
  TSBank
  GBLTracks
  RotatedHelicalOnTrackHits
  SVTRawHitsOnTrack
  SVTRawTrackerHits
  RotatedHelicalTrackHits
  RecoEcalHits
  RecoEcalClusters
  UnconstrainedV0Vertices
  ParticlesOnVertices
    HasInconsistentHash()
    track_
      HasInconsistentHash()
      tracker_hits_
      particle_
      isolation_
      n_hits_
      track_volume_
      type_
      d0_
      phi0_
      omega_
      tan_lambda_
      z0_
      chi2_
      ndf_
      track_time_
      x_at_ecal_
      y_at_ecal_
      z_at_ecal_
      lambda_kinks_
      phi_kinks_
      px_
      py_
      pz_
      id_
      charge_
      nShared_
      SharedLy0_
      SharedLy1_
```

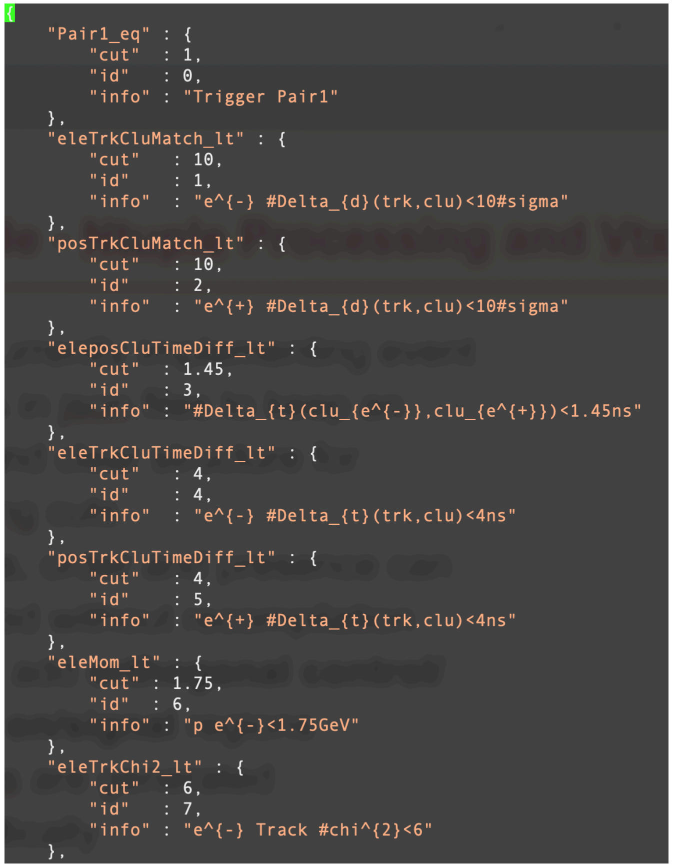


Example - Ntuple Processing and Vtx Cutflow

- Hpstr is currently implementing event selections in json files to keep an ordered and clear structure for bookeeping cuts
- Cut values, order and presence can be changed without recompilation
  - Easy to add **orthogonal control/validation/ signal** regions
  - Cutflows are generated automatically
- Vertex preselection cutflow matches between flat-tuple based analysis and hpstr based analysis => **analysis flow validated**



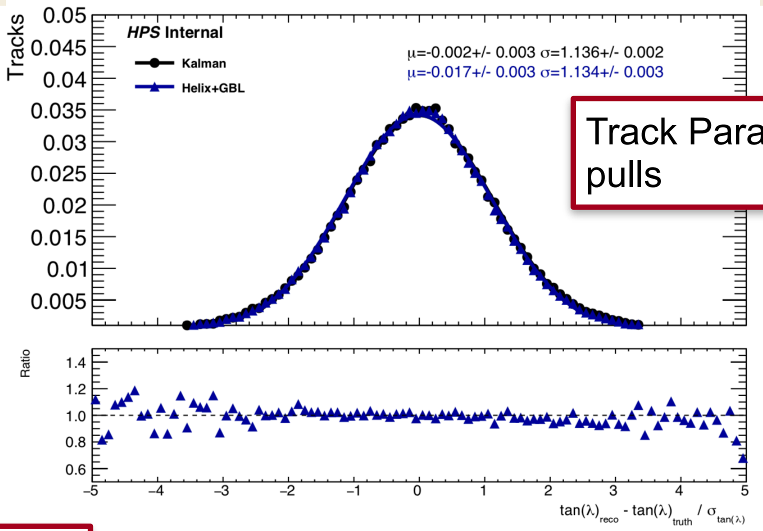
analysis/selections/vtxSelection.json


```
{
    "Pair1_eq" : {
        "cut"  : 1,
        "id"   : 0,
        "info" : "Trigger Pair1"
    },
    "eleTrkCluMatch_lt" : {
        "cut"    : 10,
        "id"     : 1,
        "info"   : "e^{-} #Delta_{d}(trk,clu)<10#sigma"
    },
    "posTrkCluMatch_lt" : {
        "cut"    : 10,
        "id"     : 2,
        "info"   : "e^{+} #Delta_{d}(trk,clu)<10#sigma"
    },
    "eleposCluTimeDiff_lt" : {
        "cut"  : 1.45,
        "id"   : 3,
        "info" : "#Delta_{t}(clu_{e^{-}},clu_{e^{+}})<1.45ns"
    },
    "eleTrkCluTimeDiff_lt" : {
        "cut"    : 4,
        "id"     : 4,
        "info"   : "e^{-} #Delta_{t}(trk,clu)<4ns"
    },
    "posTrkCluTimeDiff_lt" : {
        "cut"    : 4,
        "id"     : 5,
        "info"   : "e^{+} #Delta_{t}(trk,clu)<4ns"
    },
    "eleMom_lt" : {
        "cut" : 1.75,
        "id"  : 6,
        "info" : "p e^{-}<1.75GeV"
    },
    "eleTrkChi2_lt" : {
        "cut"  : 6,
        "id"   : 7,
        "info" : "e^{-} Track #chi^{2}<6"
    },
```

Sorry, plot on old selection, but still valid

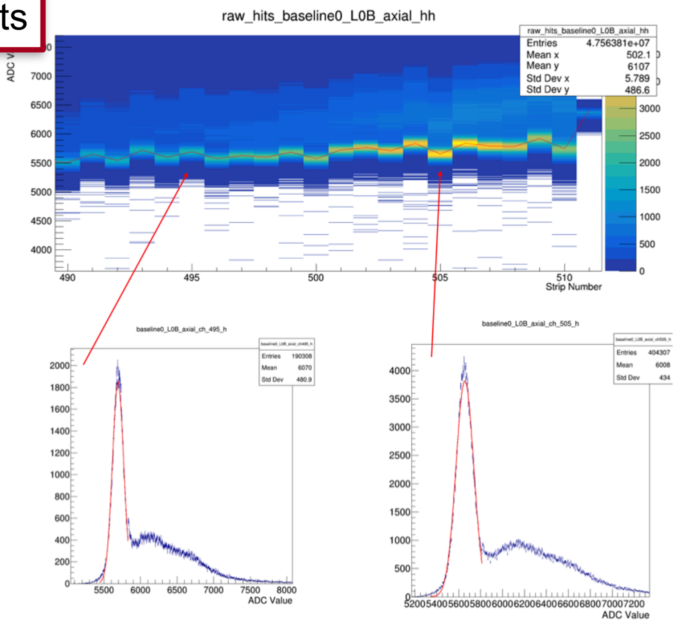
Examples - From Tracking/Hit studies to Statistical interpretation

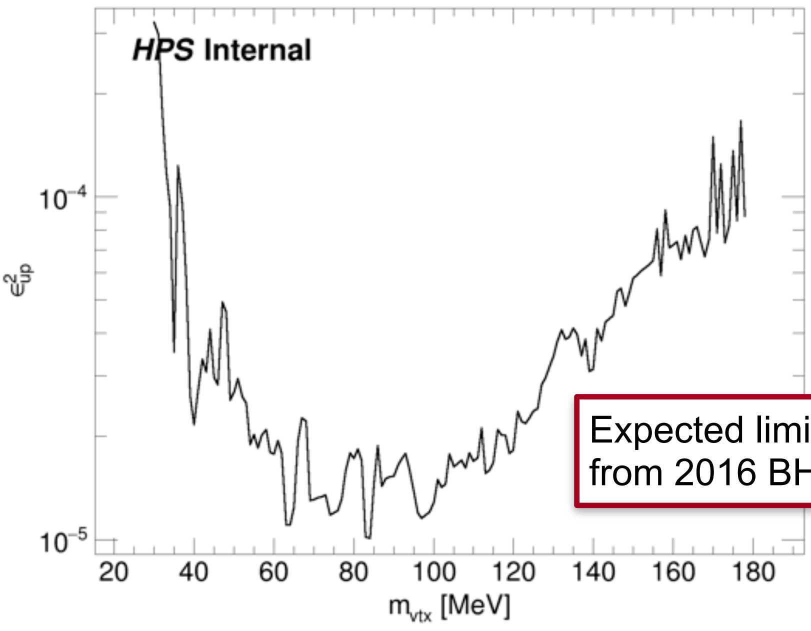
- Hpstr can provide support for also for performance studies:
  - **Tracking analysis**, i.e. Kalman-GBL comparison, track efficiency, truth matching…
  - Baseline extraction for **SVT hits** including gaussian+landau fit for charge deposition and baseline extraction
  - **Calo/Hodo hits/cluster studies**
- Of course analysis flow, radiative fraction and statistical interpretation for BH analysis are included in the framework too
- **LOT of these plots are straightforward to produce and can be easily made by others that want to help the validation/ find bugs/help producing results**



Track Params pulls

baseline fits





Expected limits (10% Lumi) from 2016 BH search

Some fast checks of tri-trig+beam MC 2019

- Cuts:

  - ElectronP < 4.5 GeV

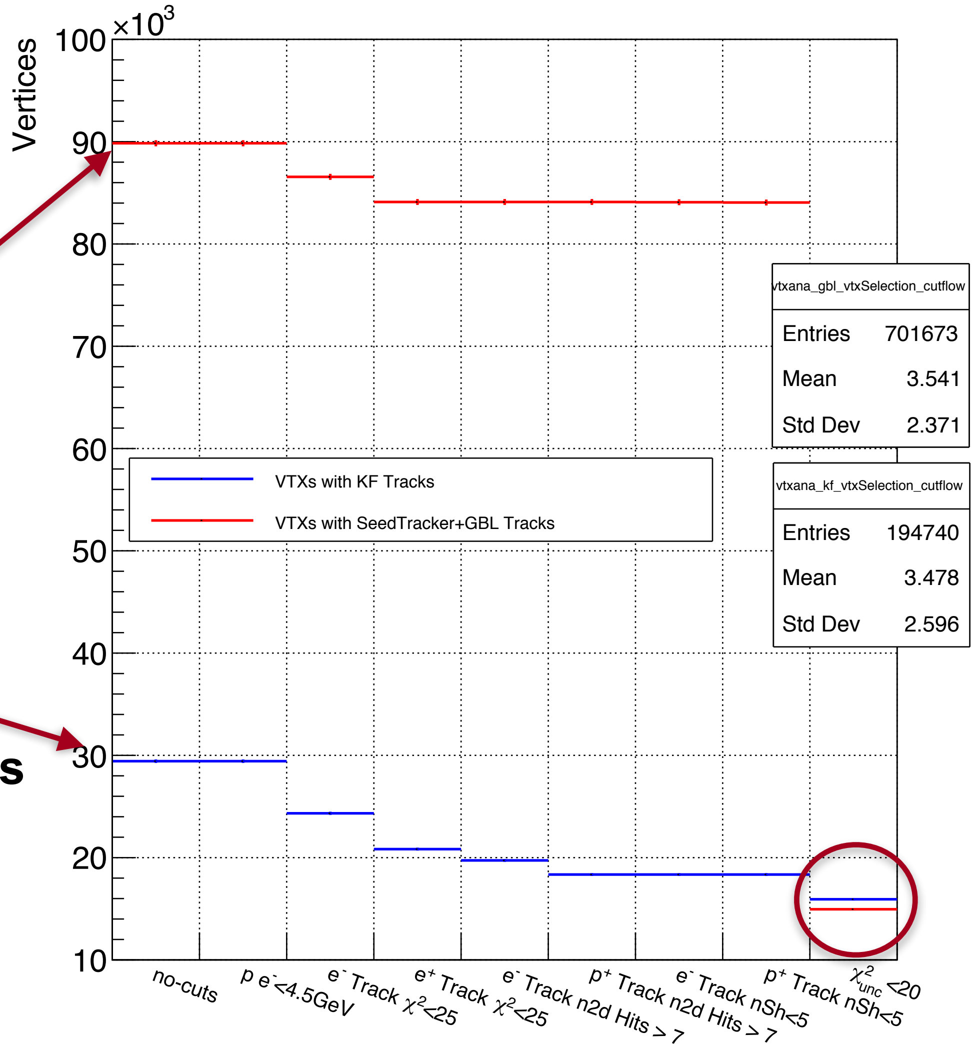
  - Ele/Pos Chi2 < 25

  - Ele/Pos n2DHits>=7 [by mistake, should have been >]

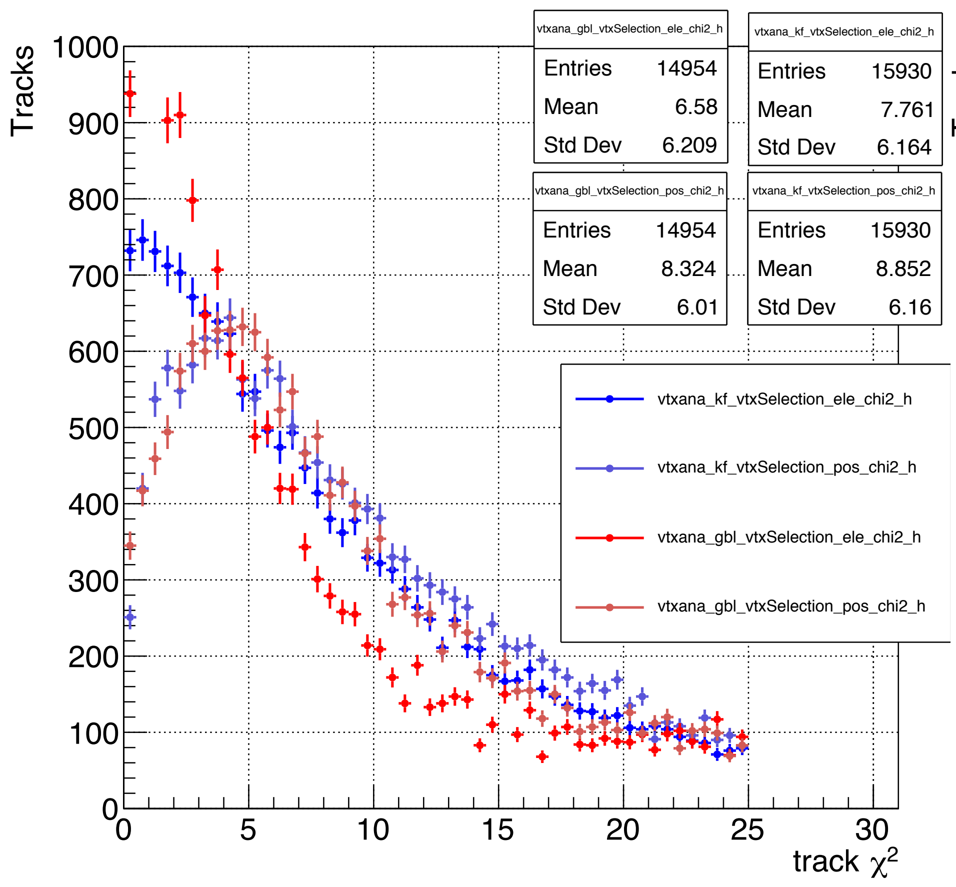
  - Ele/Pos nSharedHits<5 [no effect, due to MOUSE]

  - UncVtx Chi2<20

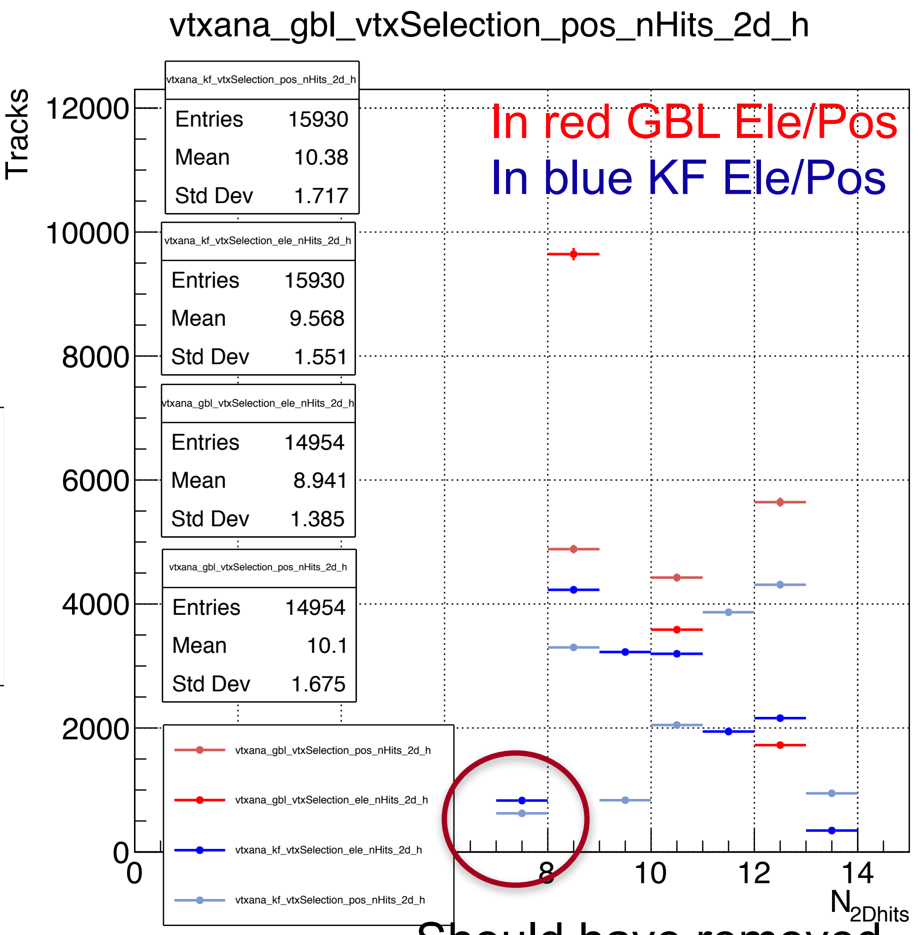
MANY more tracks with SeedTracker, wrt KF to begin with.
However: we know we have lot of lowQuality tracks and duplicates VtxChi2 cleans them all up.
I think this is in line with the long processing time of our standard tracking



Some fast checks of tri-trig+beam MC 2019



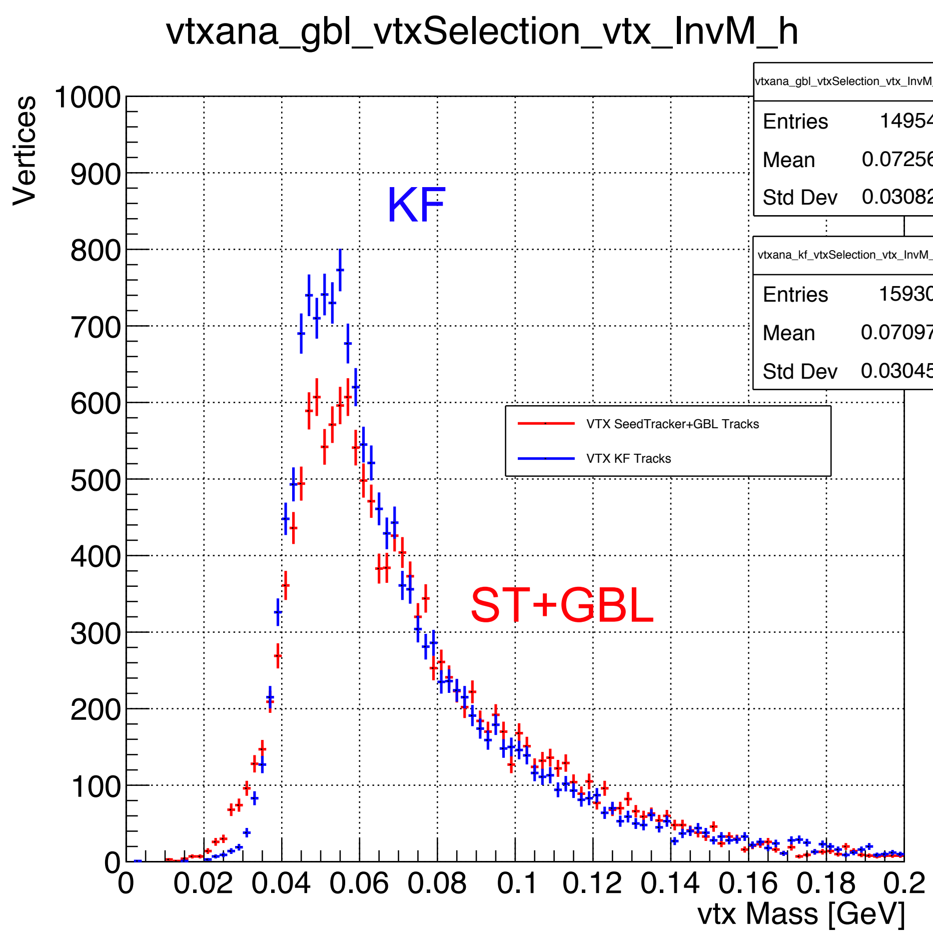
In red GBL Ele/Pos
In blue KF Ele/Pos

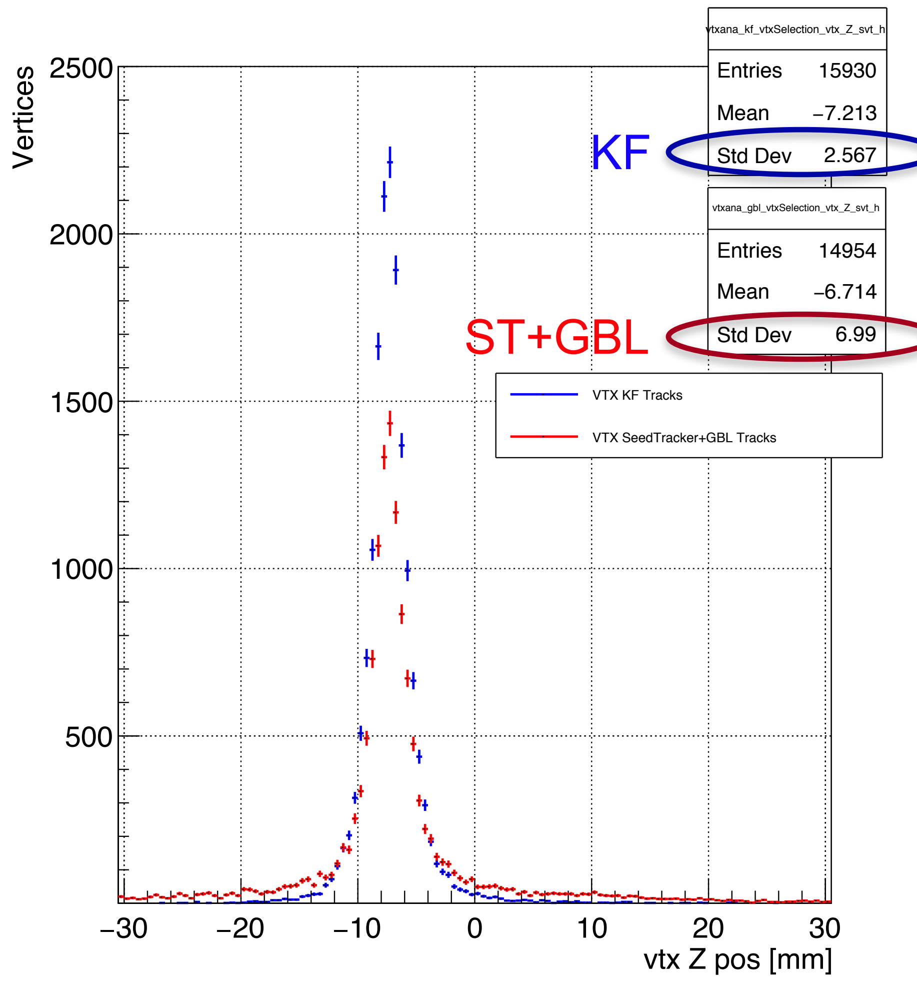


In red GBL Ele/Pos
In blue KF Ele/Pos

Should have removed those

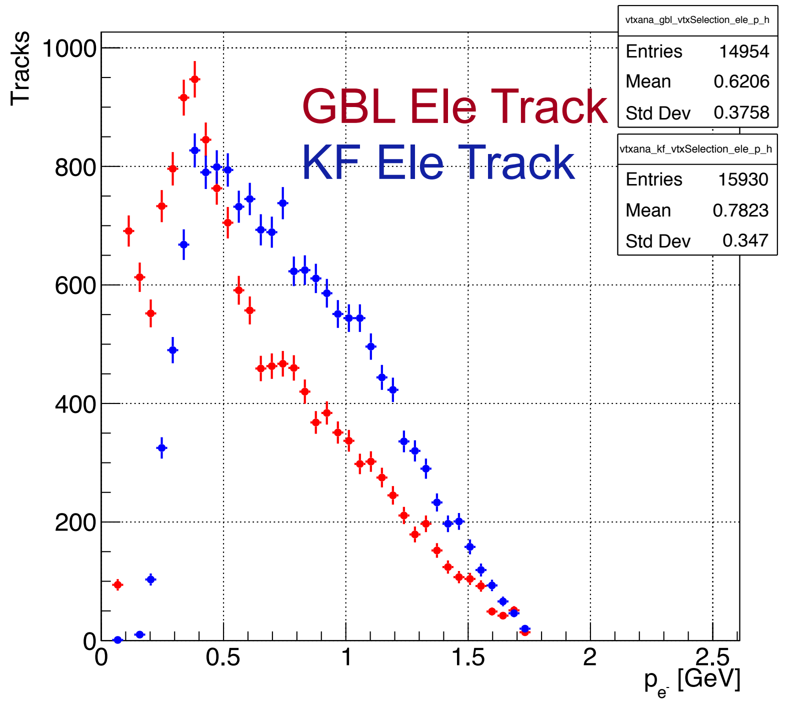
Some fast checks of tri-trig+beam MC 2019





Current issues

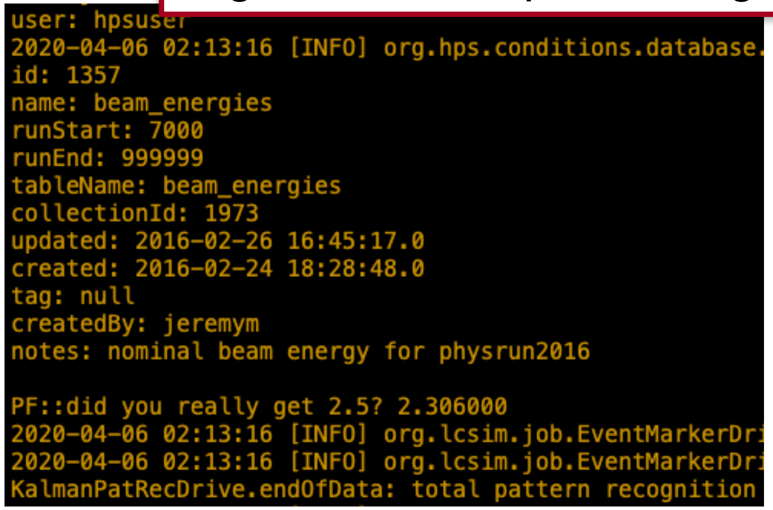
- BeamEnergy in MC samples remained set on 2016 value.
- Unfortunate combination of:
  - Old defaults in the reconstruction database
  - Defaults of the StandardCuts class that define the cuts in the ReconParticleDriver
- Effect:
  ElectronMomentum < 0.75*2.30 GeV



vtxana_gbl_vtxSelection_ele_p_h	
Entries	14954
Mean	0.6206
Std Dev	0.3758

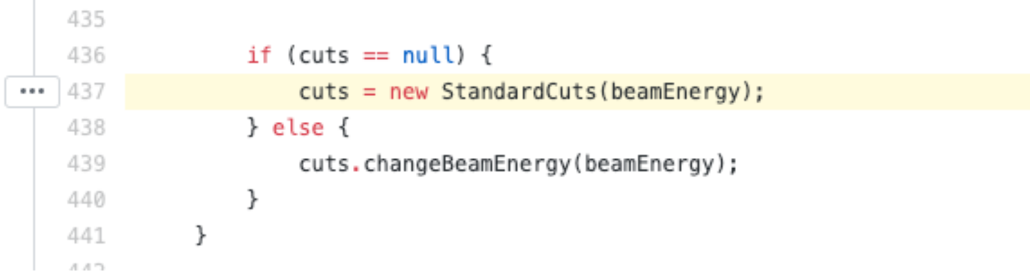
vtxana_kf_vtxSelection_ele_p_h	
Entries	15930
Mean	0.7823
Std Dev	0.347

Log from trident processing

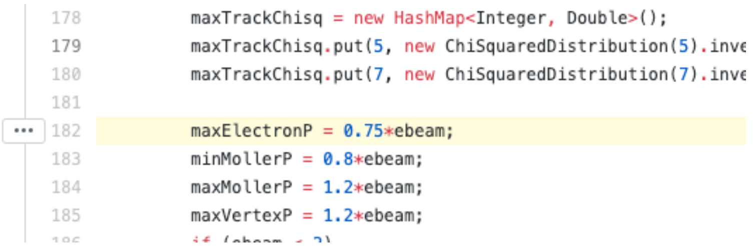


```
user: hpsuser
2020-04-06 02:13:16 [INFO] org.hps.conditions.database.
id: 1357
name: beam_energies
runStart: 7000
runEnd: 999999
tableName: beam_energies
collectionId: 1973
updated: 2016-02-26 16:45:17.0
created: 2016-02-24 18:28:48.0
tag: null
createdBy: jeremym
notes: nominal beam energy for physrun2016

PF::did you really get 2.5? 2.306000
2020-04-06 02:13:16 [INFO] org.lcsim.job.EventMarkerDri
2020-04-06 02:13:16 [INFO] org.lcsim.job.EventMarkerDri
KalmanPatRecDrive.endOfData: total pattern recognition
```



```
            if (cuts == null) {
                cuts = new StandardCuts(beamEnergy);
            } else {
                cuts.changeBeamEnergy(beamEnergy);
            }
        }
```



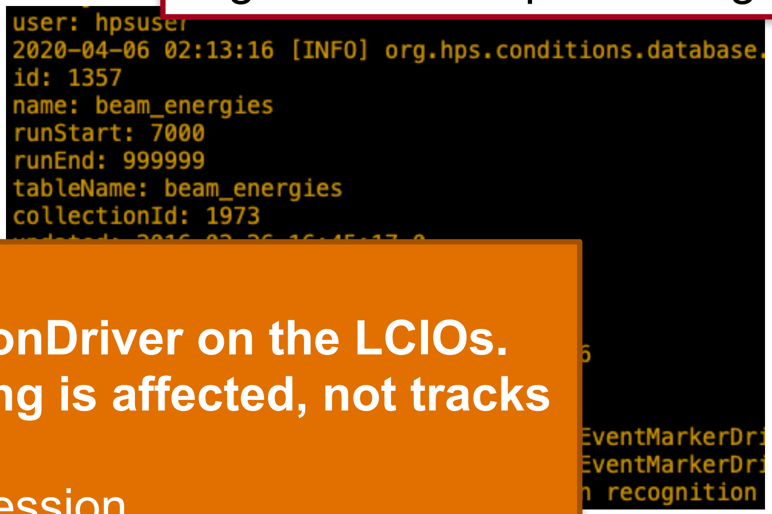
```
        maxTrackChisq = new HashMap<Integer, Double>();
        maxTrackChisq.put(5, new ChiSquaredDistribution(5).inve
        maxTrackChisq.put(7, new ChiSquaredDistribution(7).inve

        maxElectronP = 0.75*ebeam;
        minMollerP = 0.8*ebeam;
        maxMollerP = 1.2*ebeam;
        maxVertexP = 1.2*ebeam;
```

Current issues

- BeamEnergy in MC samples remained set on 2016 value.
- Unfortunate combination of:
  - Old defaults in the reconstruction database
  - Defaults of the StandardCuts class that define the cuts in the ReconParticleDriver
- Effect:
  ElectronMome

Log from trident processing



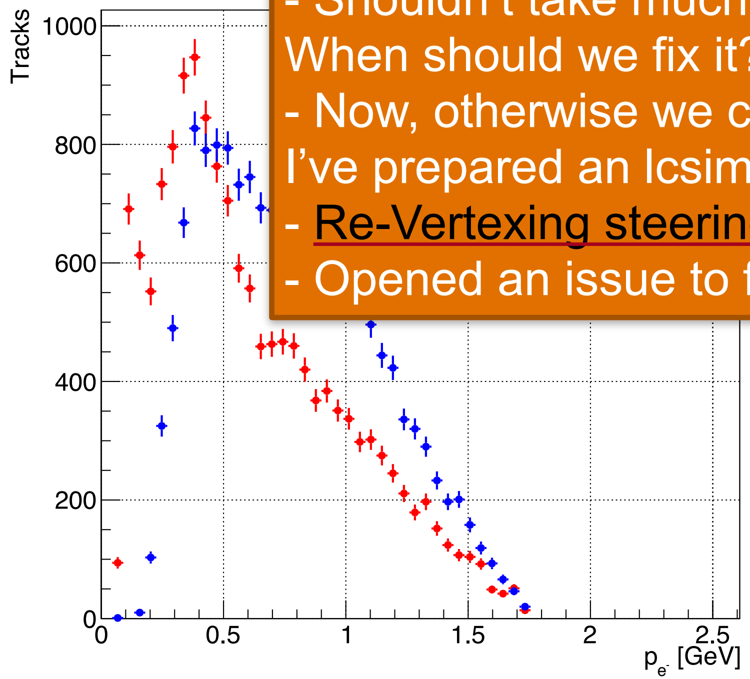
```
user: hpsuser
2020-04-06 02:13:16 [INFO] org.hps.conditions.database.
id: 1357
name: beam_energies
runStart: 7000
runEnd: 999999
tableName: beam_energies
collectionId: 1973
```

```
EventMarkerDri
EventMarkerDri
n recognition
```

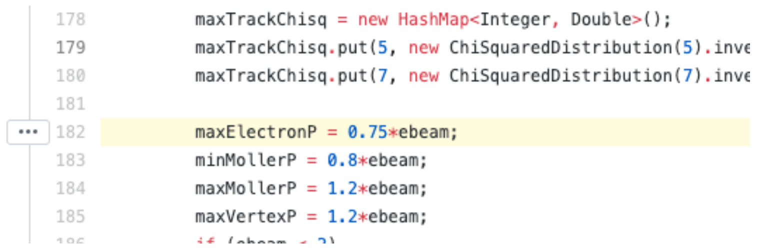
Can we fix it?
- **Yes: we just have to rerun the ReconDriver on the LCIOs.**
- Shouldn't take much as **only vertexing is affected, not tracks**

When should we fix it?
- Now, otherwise we can cancel wed session

I've prepared an lcsim steering file that "fixes" that.
- Re-Vertexing steering-file
- Opened an issue to fix DB: iss690



```
        }
    }
```



```
        maxTrackChisq = new HashMap<Integer, Double>();
        maxTrackChisq.put(5, new ChiSquaredDistribution(5).inve
        maxTrackChisq.put(7, new ChiSquaredDistribution(7).inve

        maxElectronP = 0.75*ebeam;
        minMollerP = 0.8*ebeam;
        maxMollerP = 1.2*ebeam;
        maxVertexP = 1.2*ebeam;
```

BACKUP

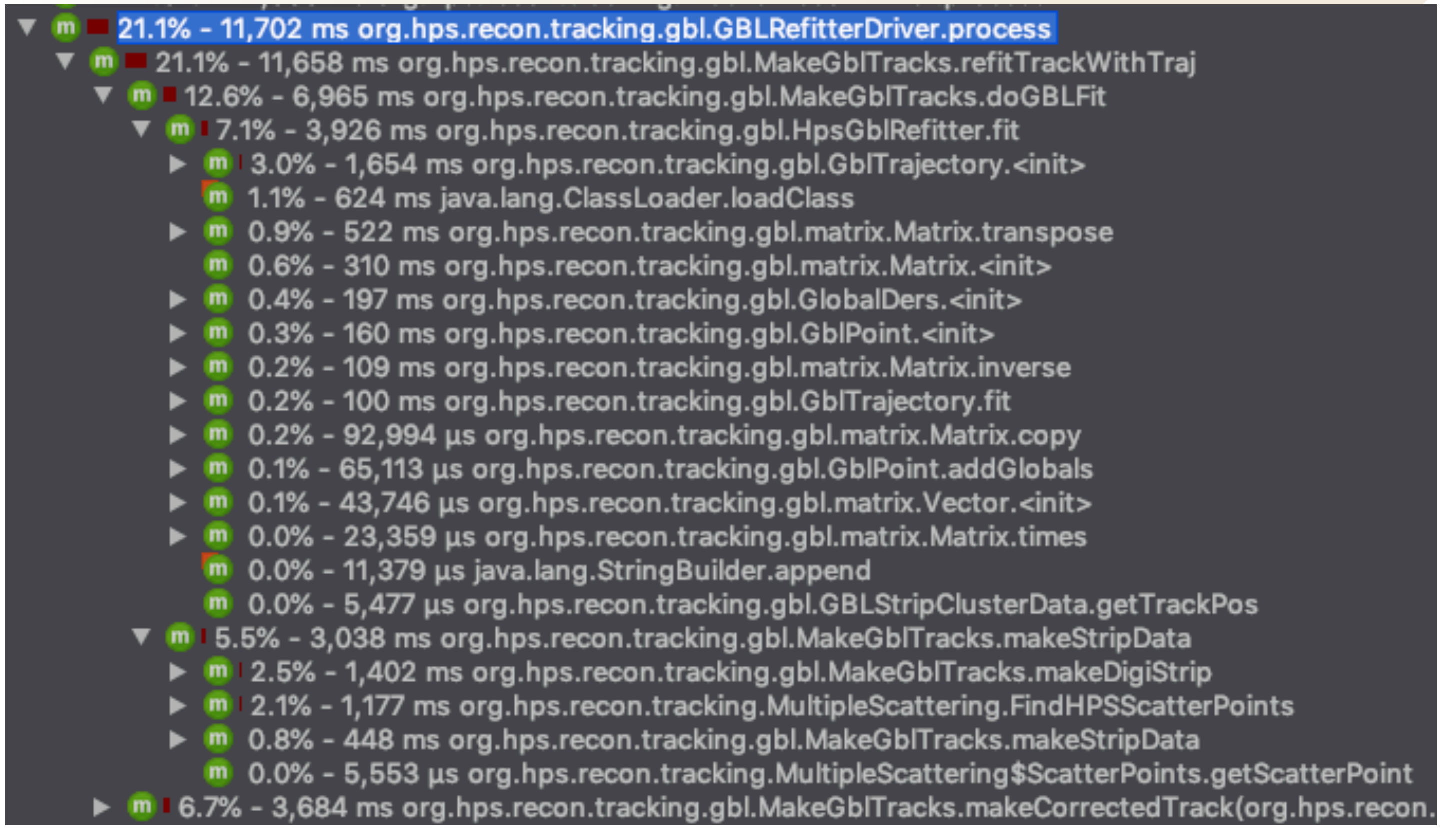
Open point for discussion - in random order

- (1) **Proper solution for Monster Events - DATA**
  - We need to fix the SVT Event Filter to remove/skip un-physical events.
  - The current Driver is tuned on 2015 - 2016 studies and need to be fixed for 2019. Current workaround limit of max 200 Clusters/event is arbitrary.
- (2) **MCParticle container in the LCIO is huge** (found about 3k MC Particles per event in the tri-trig + beam)
  - Need to apply cuts before they arrive in final LCIO files
- (3) **Tracking Processing time**
  - Current tracking strategy probably not sustainable in 2019 as takes too much processing time
  - KF pattern reco can be an alternative, once validated and when everyone's happy
- **(4) Raw Hit Fitting Time - DATA (and MC?)**
  - RawSVTHitFittingTime takes 30% of evio->LCIO step in 2019 Data. Can be partially fixed by (1).
  - Alternatively a 2 step process?
    - First we perform a EVIO->FHO [FittedHitsOnly]
    - Hand the FHO for Reconstruction/Alignment/Analysis to people.
      This will cuts 30% of processing time when we'll need to process all the data.
- (5) **BeamSpot determination from Data**
  - BeamSpot info as free parameter are dangerous if BS moves [2016 vex had to recompute it at analysis level]
- (6) **Start an event skimming campaign**
  - A non-negligible amount of events do not even have tracks in them leading to slow processing.
  - Trigger-wise or basic skimming should be done to ensure we don't spend too much time running on useless data. Better earlier than later.
- (7) **Keep track of processing commands**
  - Data we process is often private made with private steering files. We should keep track of what we did in the case of a larger official production.

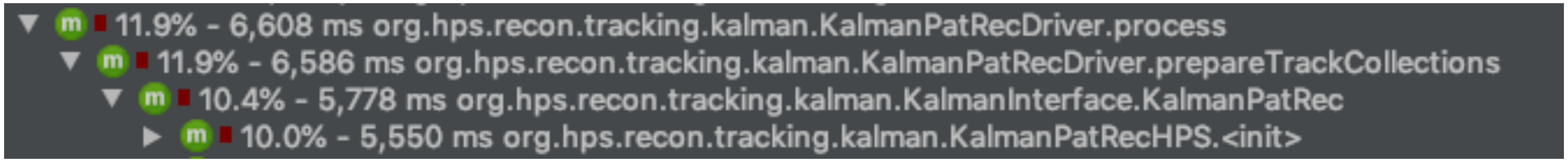
BACKUP

- **(4) Raw Hit Fitting Time**
  - RawSVTHitFittingTime takes 30% of evio->LCIO step in 2019 Data. Can be partially fixed by (1).
  - A possible compromise while we develop a faster fitting machinery is to reconstruct data in 2 steps:
    - First we perform a EVIO->FHO on the files we want/need [FittedHitsOnly LCIO files] and could start basically today.
    - Use the FHO for Reconstruction/Alignment/Analysis. This will cuts 30% of processing time when we'll need to process all the data.
  - If eventually we get to change fitting we can restart the chain
  - Not directly LCIO as quite slow at the moment and LCIO ntuples content might will change soon.
- (5) **BeamSpot determination from Data**
  - BeamSpot info as free parameter are dangerous if BS moves [2016 vex had to recompute it at analysis level]
  - Propose to do a double processing: x-process and f-process
  - x-process to compute BS information (position/sigma) and store in DB, then f-process for proper correct event-by-event BS/Target constraint.

jProfiler on tri-trig without beam bkg

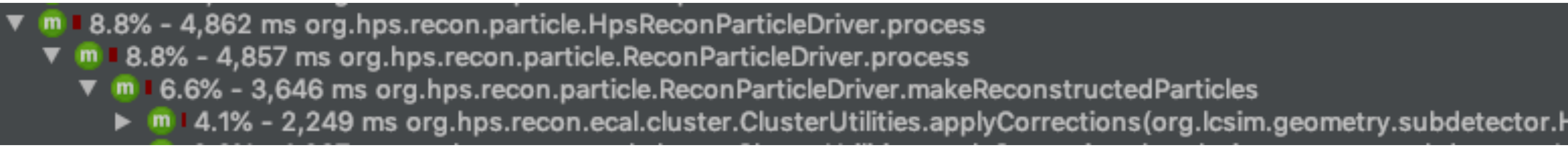


```
▼ 21.1% - 11,702 ms org.hps.recon.tracking.gbl.GBLRefitterDriver.process
  ▼ 21.1% - 11,658 ms org.hps.recon.tracking.gbl.MakeGblTracks.refitTrackWithTraj
    ▼ 12.6% - 6,965 ms org.hps.recon.tracking.gbl.MakeGblTracks.doGBLFit
      ▼ 7.1% - 3,926 ms org.hps.recon.tracking.gbl.HpsGblRefitter.fit
        ▶ 3.0% - 1,654 ms org.hps.recon.tracking.gbl.GblTrajectory.<init>
          1.1% - 624 ms java.lang.ClassLoader.loadClass
        ▶ 0.9% - 522 ms org.hps.recon.tracking.gbl.matrix.Matrix.transpose
          0.6% - 310 ms org.hps.recon.tracking.gbl.matrix.Matrix.<init>
        ▶ 0.4% - 197 ms org.hps.recon.tracking.gbl.GlobalDers.<init>
        ▶ 0.3% - 160 ms org.hps.recon.tracking.gbl.GblPoint.<init>
        ▶ 0.2% - 109 ms org.hps.recon.tracking.gbl.matrix.Matrix.inverse
        ▶ 0.2% - 100 ms org.hps.recon.tracking.gbl.GblTrajectory.fit
        ▶ 0.2% - 92,994 µs org.hps.recon.tracking.gbl.matrix.Matrix.copy
        ▶ 0.1% - 65,113 µs org.hps.recon.tracking.gbl.GblPoint.addGlobals
        ▶ 0.1% - 43,746 µs org.hps.recon.tracking.gbl.matrix.Vector.<init>
        ▶ 0.0% - 23,359 µs org.hps.recon.tracking.gbl.matrix.Matrix.times
          0.0% - 11,379 µs java.lang.StringBuilder.append
          0.0% - 5,477 µs org.hps.recon.tracking.gbl.GBLStripClusterData.getTrackPos
      ▼ 5.5% - 3,038 ms org.hps.recon.tracking.gbl.MakeGblTracks.makeStripData
        ▶ 2.5% - 1,402 ms org.hps.recon.tracking.gbl.MakeGblTracks.makeDigiStrip
        ▶ 2.1% - 1,177 ms org.hps.recon.tracking.MultipleScattering.FindHPSScatterPoints
        ▶ 0.8% - 448 ms org.hps.recon.tracking.gbl.MakeGblTracks.makeStripData
          0.0% - 5,553 µs org.hps.recon.tracking.MultipleScattering$ScatterPoints.getScatterPoint
    ▶ 6.7% - 3,684 ms org.hps.recon.tracking.gbl.MakeGblTracks.makeCorrectedTrack(org.hps.recon.
```

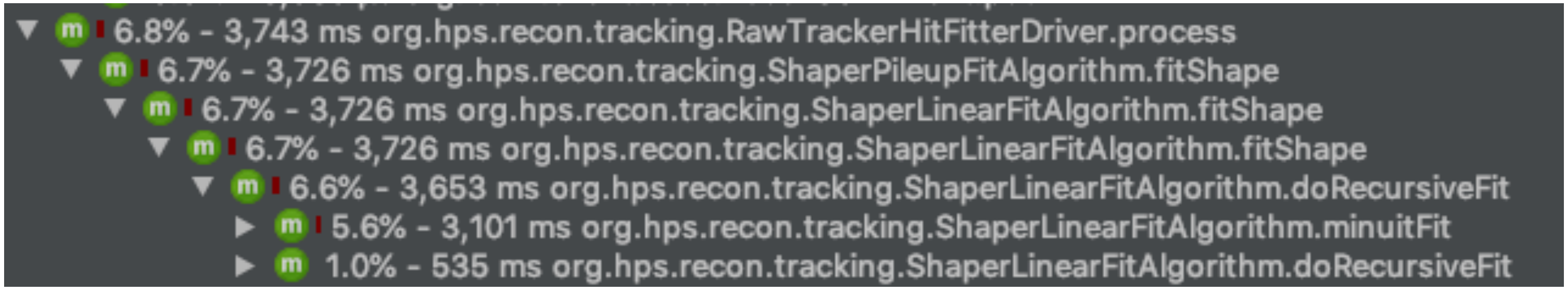


```
▼ 11.9% - 6,608 ms org.hps.recon.tracking.kalman.KalmanPatRecDriver.process
  ▼ 11.9% - 6,586 ms org.hps.recon.tracking.kalman.KalmanPatRecDriver.prepareTrackCollections
    ▼ 10.4% - 5,778 ms org.hps.recon.tracking.kalman.KalmanInterface.KalmanPatRec
      ▶ 10.0% - 5,550 ms org.hps.recon.tracking.kalman.KalmanPatRecHPS.<init>
```

jProfiler on tri-trig without beam bkg

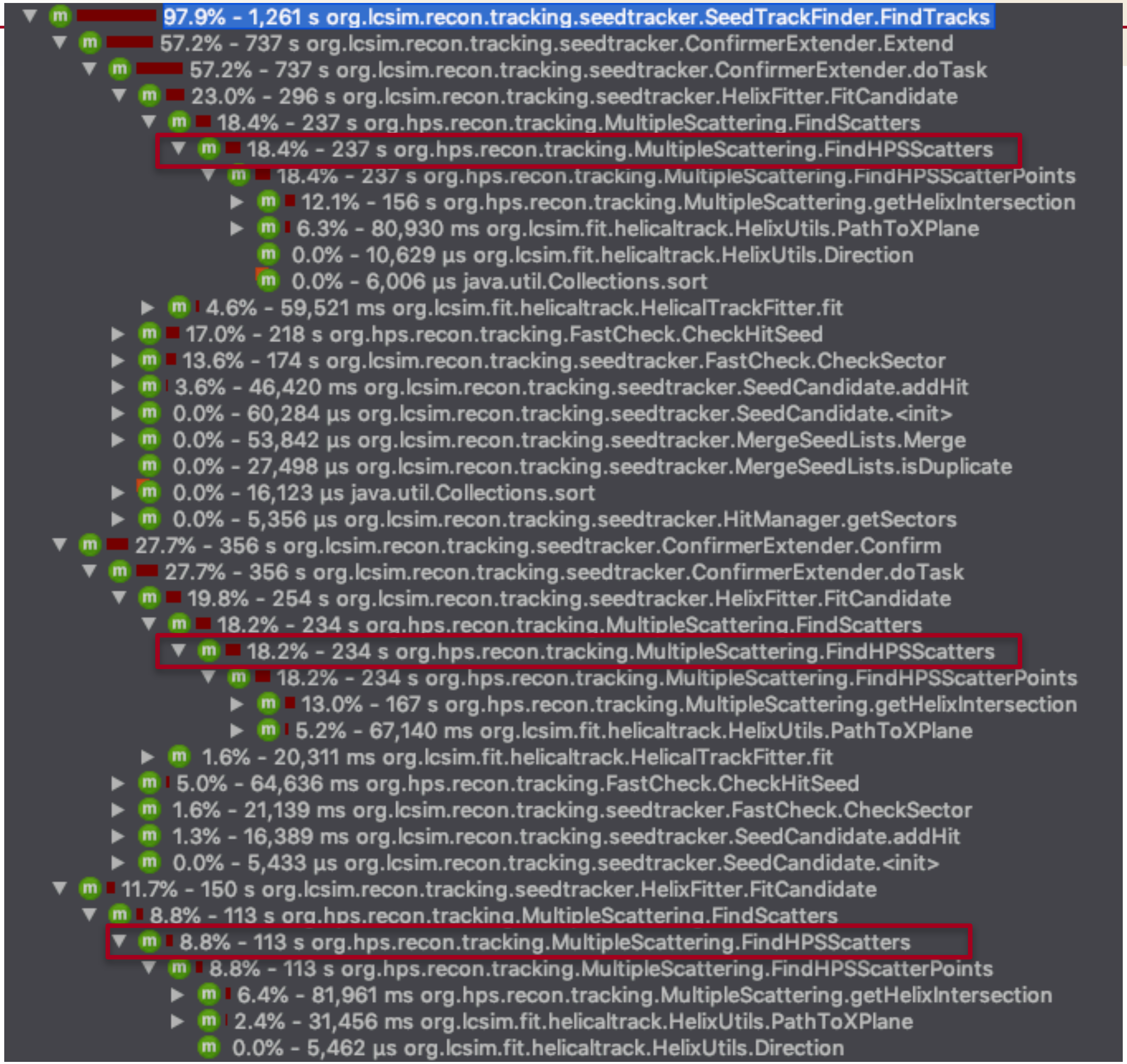


```
▼ 8.8% - 4,862 ms org.hps.recon.particle.HpsReconParticleDriver.process
  ▼ 8.8% - 4,857 ms org.hps.recon.particle.ReconParticleDriver.process
    ▼ 6.6% - 3,646 ms org.hps.recon.particle.ReconParticleDriver.makeReconstructedParticles
      ▶ 4.1% - 2,249 ms org.hps.recon.ecal.cluster.ClusterUtilities.applyCorrections(org.lcsim.geometry.subdetector.H
```



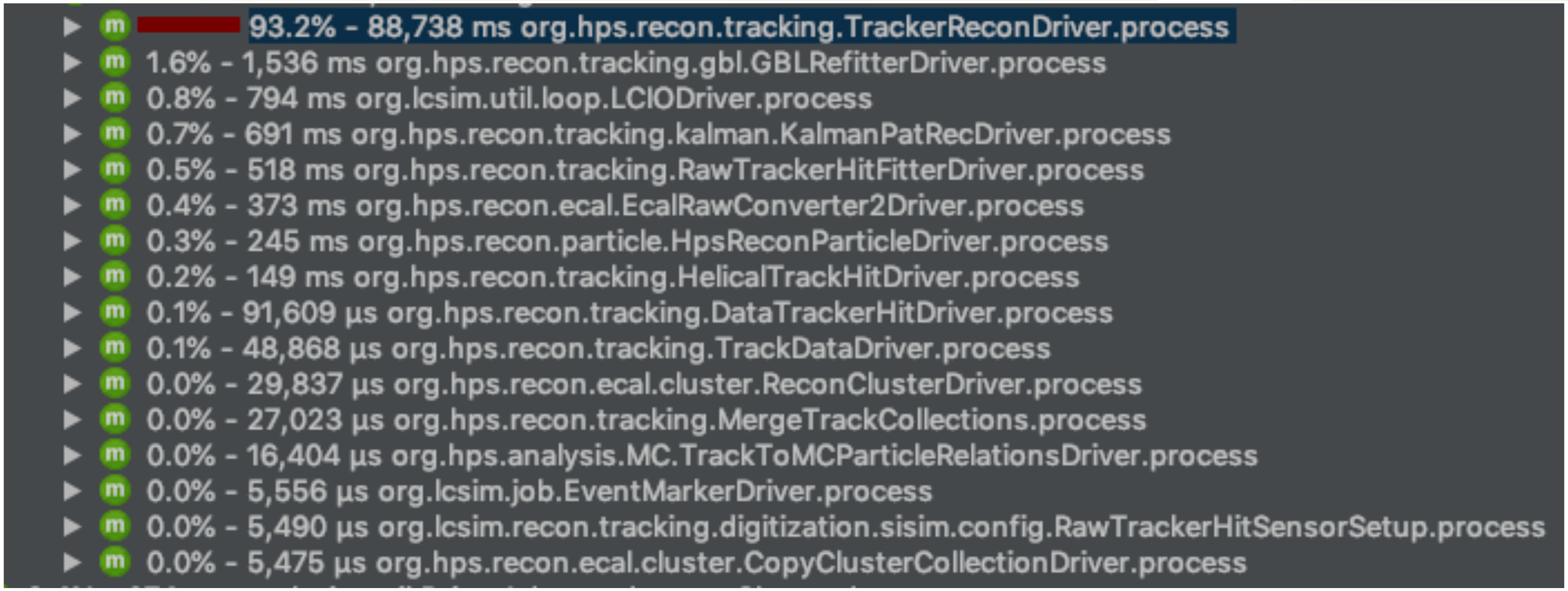
```
▼ 6.8% - 3,743 ms org.hps.recon.tracking.RawTrackerHitFitterDriver.process
  ▼ 6.7% - 3,726 ms org.hps.recon.tracking.ShaperPileupFitAlgorithm.fitShape
    ▼ 6.7% - 3,726 ms org.hps.recon.tracking.ShaperLinearFitAlgorithm.fitShape
      ▼ 6.7% - 3,726 ms org.hps.recon.tracking.ShaperLinearFitAlgorithm.fitShape
        ▼ 6.6% - 3,653 ms org.hps.recon.tracking.ShaperLinearFitAlgorithm.doRecursiveFit
          ▶ 5.6% - 3,101 ms org.hps.recon.tracking.ShaperLinearFitAlgorithm.minuitFit
          ▶ 1.0% - 535 ms org.hps.recon.tracking.ShaperLinearFitAlgorithm.doRecursiveFit
```

jProfiler on tri-trig with beam bkg



```
▼ 97.9% - 1,261 s org.lcsim.recon.tracking.seedtracker.SeedTrackFinder.FindTracks
  ▼ 57.2% - 737 s org.lcsim.recon.tracking.seedtracker.ConfirmerExtender.Extend
    ▼ 57.2% - 737 s org.lcsim.recon.tracking.seedtracker.ConfirmerExtender.doTask
      ▼ 23.0% - 296 s org.lcsim.recon.tracking.seedtracker.HelixFitter.FitCandidate
        ▼ 18.4% - 237 s org.hps.recon.tracking.MultipleScattering.FindScatters
          ▼ 18.4% - 237 s org.hps.recon.tracking.MultipleScattering.FindHPSScatters
            ▼ 18.4% - 237 s org.hps.recon.tracking.MultipleScattering.FindHPSScatterPoints
              ▶ 12.1% - 156 s org.hps.recon.tracking.MultipleScattering.getHelixIntersection
              ▶ 6.3% - 80,930 ms org.lcsim.fit.helicaltrack.HelixUtils.PathToXPlane
                0.0% - 10,629 µs org.lcsim.fit.helicaltrack.HelixUtils.Direction
                0.0% - 6,006 µs java.util.Collections.sort
        ▶ 4.6% - 59,521 ms org.lcsim.fit.helicaltrack.HelicalTrackFitter.fit
      ▶ 17.0% - 218 s org.hps.recon.tracking.FastCheck.CheckHitSeed
      ▶ 13.6% - 174 s org.lcsim.recon.tracking.seedtracker.FastCheck.CheckSector
      ▶ 3.6% - 46,420 ms org.lcsim.recon.tracking.seedtracker.SeedCandidate.addHit
      ▶ 0.0% - 60,284 µs org.lcsim.recon.tracking.seedtracker.SeedCandidate.<init>
      ▶ 0.0% - 53,842 µs org.lcsim.recon.tracking.seedtracker.MergeSeedLists.Merge
        0.0% - 27,498 µs org.lcsim.recon.tracking.seedtracker.MergeSeedLists.isDuplicate
      ▶ 0.0% - 16,123 µs java.util.Collections.sort
      ▶ 0.0% - 5,356 µs org.lcsim.recon.tracking.seedtracker.HitManager.getSectors
  ▼ 27.7% - 356 s org.lcsim.recon.tracking.seedtracker.ConfirmerExtender.Confirm
    ▼ 27.7% - 356 s org.lcsim.recon.tracking.seedtracker.ConfirmerExtender.doTask
      ▼ 19.8% - 254 s org.lcsim.recon.tracking.seedtracker.HelixFitter.FitCandidate
        ▼ 18.2% - 234 s org.hps.recon.tracking.MultipleScattering.FindScatters
          ▼ 18.2% - 234 s org.hps.recon.tracking.MultipleScattering.FindHPSScatters
            ▼ 18.2% - 234 s org.hps.recon.tracking.MultipleScattering.FindHPSScatterPoints
              ▶ 13.0% - 167 s org.hps.recon.tracking.MultipleScattering.getHelixIntersection
              ▶ 5.2% - 67,140 ms org.lcsim.fit.helicaltrack.HelixUtils.PathToXPlane
        ▶ 1.6% - 20,311 ms org.lcsim.fit.helicaltrack.HelicalTrackFitter.fit
      ▶ 5.0% - 64,636 ms org.hps.recon.tracking.FastCheck.CheckHitSeed
      ▶ 1.6% - 21,139 ms org.lcsim.recon.tracking.seedtracker.FastCheck.CheckSector
      ▶ 1.3% - 16,389 ms org.lcsim.recon.tracking.seedtracker.SeedCandidate.addHit
      ▶ 0.0% - 5,433 µs org.lcsim.recon.tracking.seedtracker.SeedCandidate.<init>
  ▼ 11.7% - 150 s org.lcsim.recon.tracking.seedtracker.HelixFitter.FitCandidate
    ▼ 8.8% - 113 s org.hps.recon.tracking.MultipleScattering.FindScatters
      ▼ 8.8% - 113 s org.hps.recon.tracking.MultipleScattering.FindHPSScatters
        ▼ 8.8% - 113 s org.hps.recon.tracking.MultipleScattering.FindHPSScatterPoints
          ▶ 6.4% - 81,961 ms org.hps.recon.tracking.MultipleScattering.getHelixIntersection
          ▶ 2.4% - 31,456 ms org.lcsim.fit.helicaltrack.HelixUtils.PathToXPlane
            0.0% - 5,462 µs org.lcsim.fit.helicaltrack.HelixUtils.Direction
```

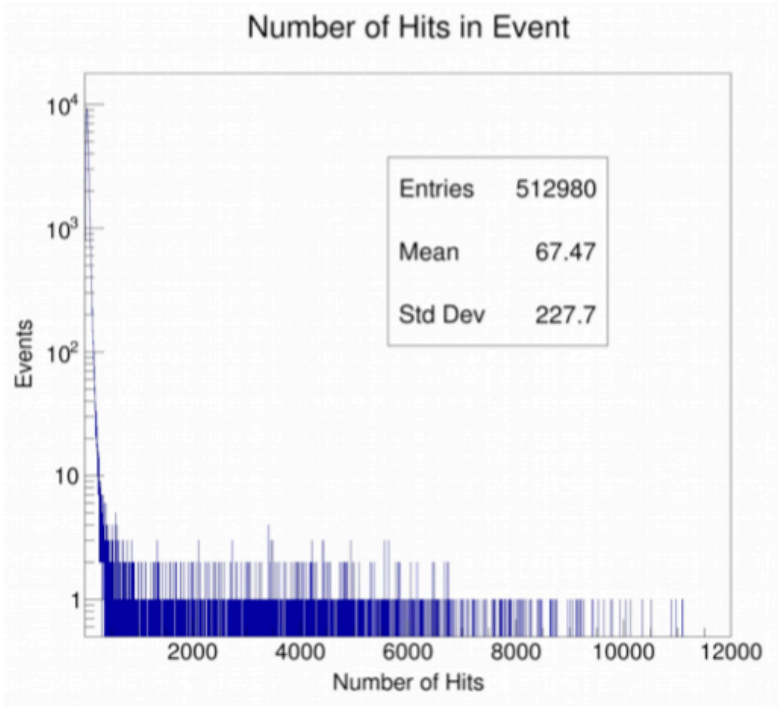
jProfiler on tri-trig with beam bkg - rmsTimeCut = 20



```
93.2% - 88,738 ms org.hps.recon.tracking.TrackerReconDriver.process
1.6% - 1,536 ms org.hps.recon.tracking.gbl.GBLRefitterDriver.process
0.8% - 794 ms org.lcsim.util.loop.LCIODriver.process
0.7% - 691 ms org.hps.recon.tracking.kalman.KalmanPatRecDriver.process
0.5% - 518 ms org.hps.recon.tracking.RawTrackerHitFitterDriver.process
0.4% - 373 ms org.hps.recon.ecal.EcalRawConverter2Driver.process
0.3% - 245 ms org.hps.recon.particle.HpsReconParticleDriver.process
0.2% - 149 ms org.hps.recon.tracking.HelicalTrackHitDriver.process
0.1% - 91,609 µs org.hps.recon.tracking.DataTrackerHitDriver.process
0.1% - 48,868 µs org.hps.recon.tracking.TrackDataDriver.process
0.0% - 29,837 µs org.hps.recon.ecal.cluster.ReconClusterDriver.process
0.0% - 27,023 µs org.hps.recon.tracking.MergeTrackCollections.process
0.0% - 16,404 µs org.hps.analysis.MC.TrackToMCParticleRelationsDriver.process
0.0% - 5,556 µs org.lcsim.job.EventMarkerDriver.process
0.0% - 5,490 µs org.lcsim.recon.tracking.digitization.sisim.config.RawTrackerHitSensorSetup.process
0.0% - 5,475 µs org.hps.recon.ecal.cluster.CopyClusterCollectionDriver.process
```

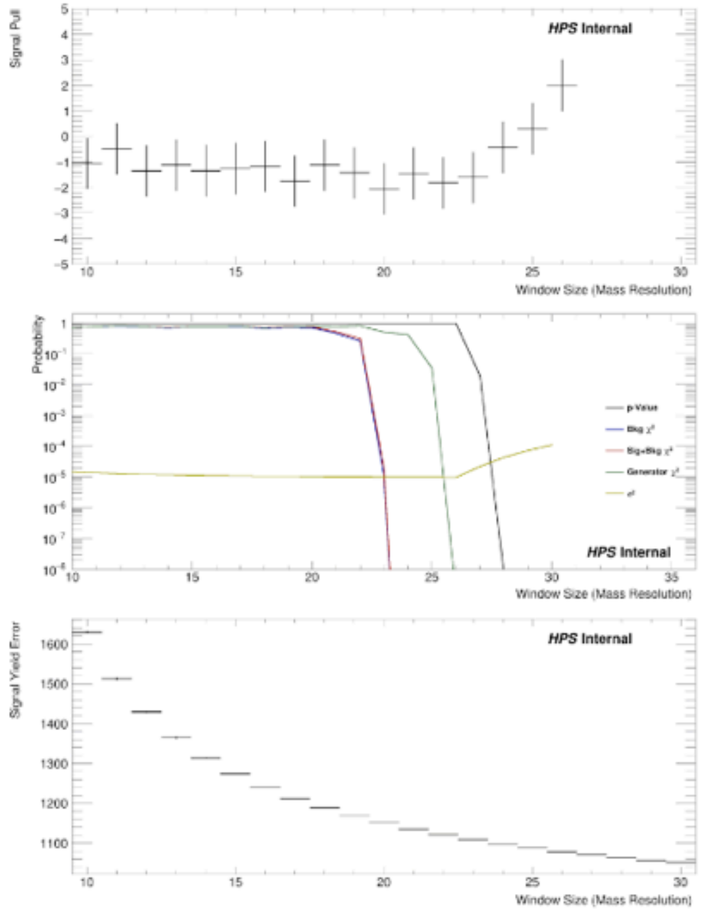
hps-java master issues when running on data

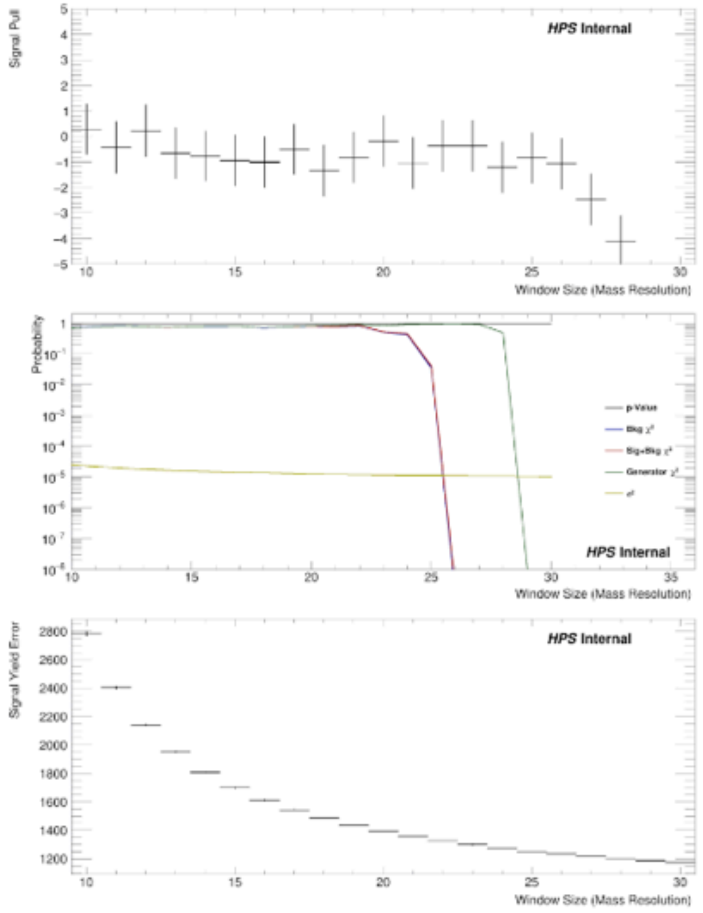
- Monster Events:
  - Order of ~% of the events have a huge amount of hits confusing the Track Finding stage
- These event are impossible to process (some lead to more than 10^3-10^4 trackCandidates)
- Current solution
  - Added protection in TrackerHitDriver for SiClusters > 200 [temporary]
  - Added configurable protection on size of **SiClusters** in **KalmanPatDriver** (Same solution of the SeedTracker)



Resonance Search Statistics Support

- Fit of mass spectrum and toy model MC with fits fully supported
- Plots for selecting bkg models available


HPS Internal
Signal Pull
Window Size (Mass Resolution)
Probability
p-Value
Bkg χ^2
Sig+Bkg χ^2
Generator χ^2
ϵ^2
Signal Yield Error


HPS Internal
Signal Pull
Window Size (Mass Resolution)
Probability
p-Value
Bkg χ^2
Sig+Bkg χ^2
Generator χ^2
ϵ^2
Signal Yield Error