SIMP Reach Estimates and Beginning Analysis

Analysis Workshop 10/19/22
Alic Spellman

Introduction

- SIMP theory and expected signal calculation overview
- New combined reach estimate plots using code from [1] **Cosmology and accelerator tests of strongly interacting dark matter**. A. Berlin, N. Blinov, S. Gori et al.
- Would like to move towards getting plots officially approved
- Started SIMPs analysis a few weeks back (progress was delayed)
- Found de-correlation between Track and matched Cluster timing that was not present in 2016 pass4 recon
- Started digging into Track/StripCluster/RawHit timing...ultimately not a serious issue and will press forward with analysis

Combined SIMP Reach Estimates
(2016+2019+2021 Runs)

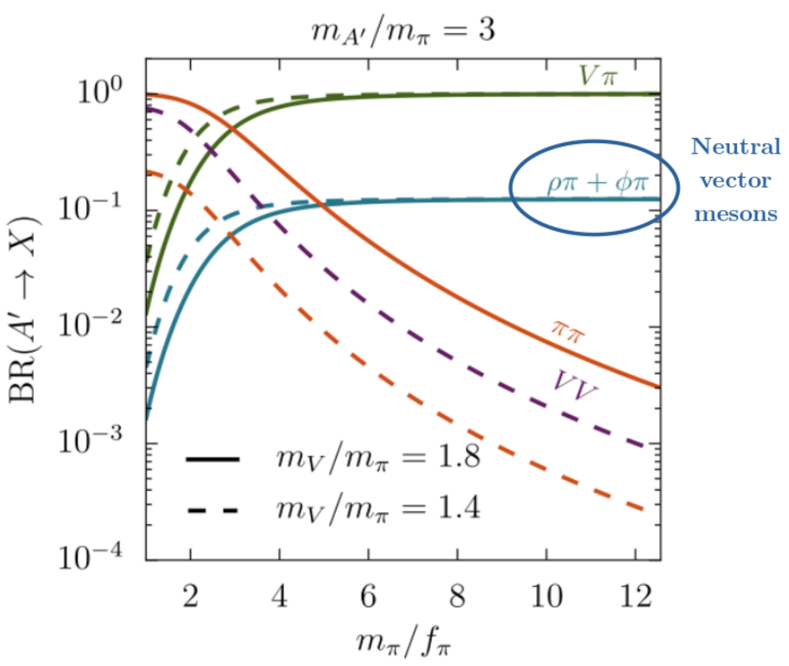
SIMP Theory Overview

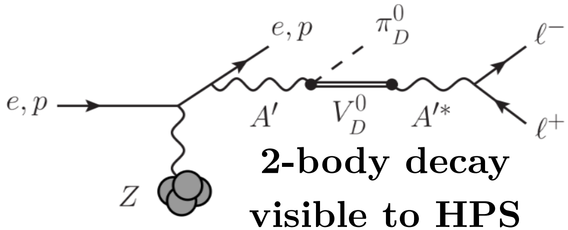
- SIMP model introduces dark sector mesons (analogous to SM) that couple to A'
- HPS is sensitive to A' prompt decay A'→ $V_D\pi_D$
  - π_D is invisible
  - Two of the five V_D states (spin-1 neutral vector mesons **ρ_D and φ_D**) are visible to HPS via 2-body decay $V_D \rightarrow e^+e^-$

- SIMP model parameters: **$m_{\pi D}$ m_{VD} $m_{\pi D}/f_{\pi D}$ $m_{A'}$ ε α_D**
- Our reach estimate follows two benchmark cases and enforces mass ratios **$m_{A'}/m_\pi = 3$, $m_V/m_\pi = 1.8$**, and **$\alpha_D = 0.1$**
- Benchmark cases fix $m_{\pi D}/f_{\pi D} = 4\pi$ and $m_{\pi D}/f_{\pi D} = 3$

- **For detailed review of SIMPs Reach Estimates, see talk here**
  - https://confluence.slac.stanford.edu/pages/viewpage.action?pageId=349284806

Branching fraction of A' into hidden sector mesons as function of ratio m_π/f_π for $m_{A'}/m_\pi = 3$, $m_V/m_\pi = 1.8$, $\varepsilon << 1$





2-body decay visible to HPS

SIMP Expected Signal Calculation

- **A' cross section** for a particular $m_{A'}$ is **related to the differential cross-section of radiative tridents** evaluated at the A' mass

$$\sigma_{A'} = \frac{3\pi m_{A'}\epsilon^2}{2N_{eff}\alpha}\frac{d\sigma_{\gamma^*}}{dm_{l^+l^-}}|_{m_{l^+l^-}=m_{A'}}$$

- Multiplying by luminosity on both sides give the **total generated A' rate**

$$N_{A'} = \frac{3\pi m_{A'}\epsilon^2}{2N_{eff}\alpha}\boxed{\frac{dN_{\gamma^*}}{dm_{A'}}}$$

Radiative tridents are not real process
However can be estimated via MC

- Expected SIMP Signal Rate calculated by applying signal branching ratios and prompt/displaced acceptance terms to total generated A' rate

$$\boxed{N_{A'sig} = N_{A'}\Theta A_{V_D} E_{vtx}(\epsilon^2)}$$

$\mathbf{A_{VD}}$ is A' signal acceptance for **prompt $\mathbf{V_D}$ decay**

$$\Theta = BR(A' \to \pi_D V_D)BR(V_D \to e^+e^-)$$

Fraction of final selected V_D vertices as function of truth vertex z position

$$E_{vtx}(\epsilon^2) = \int_{z_{tar}}^{\infty} \frac{exp^{\left(\frac{z_{target}-z}{\gamma c\tau}\right)}}{\gamma c\tau}\frac{F(z)}{A_{V_D}}dz$$

Accounts for displaced V_D decay acceptance

SIMP Expected Signal Calculation Continued

- Evaluate the **total differential radiative trident rate** for a given A' mass **inside Control Region**
- This rate is proportional to the total number of generated A's of $m_{A'}$

$$\frac{dN_{\gamma^*}}{dm_{A'}} = \boxed{\left(\frac{dN_{\gamma^*}}{dm_{A'}} \Big/ \frac{dN_{\gamma^* CR}}{dm_{A'}}\right)} \boxed{\left(\frac{dN_{\gamma^* CR}}{dm_{A'}} \Big/ \frac{dN_{CR}}{dm_{reco}}\right)} \boxed{\frac{dN_{CR}}{dm_{reco}}} \longrightarrow N_{A'} = \frac{3\pi m_{A'}\epsilon^2 \boxed{f_{rad}}}{2N_{eff}\alpha \boxed{\zeta}} \boxed{\frac{dN_{CR}}{dm_{reco}}}$$

- Each of the three terms (colored boxes) are **measured in MC**
- "Radiative Fraction" (f_{rad})
  - ratio of **selected radiative tridents to reconstructed background** vertices in **Control Region**
  - **measured purely in MC**
- **"Total Radiative Acceptance" (ζ)**
  - ratio of **selected to generated radiative tridents** in **Control Region**
  - **measured purely in MC**
- **"Reconstructed Background Rate"**
  - number of **Wab+Tritrig vertices that pass selection** in the **Control Region**
  - **Estimated in MC using approximate Lumi** for reach estimate
  - Also **measurable in data**

Expected A' Signal Rate

$$\boxed{N_{A'sig} = N_{A'}\Theta A_{V_D} E_{vtx}(\epsilon^2)}$$

SIMP Expected Signal Calculation Continued

- For each A' mass, calculate total generated A' rate using radiative fraction, total radiative acceptance, and background rate
- Measure the V_D Vertex Selection Efficiency F(z) for the V_D mass corresponding to the A' mass

Two neutral Dark Vectors (ρ and φ) contribute to Expected Signal

- The rate of A' → neutral V_D + π_D is different for ρ and φ
- The lifetimes of ρ and φ are different
- A' → Neutral V_D rate is different for ρ and φ

$$\Gamma(A' \to V_D \pi_D) = \frac{\alpha_D T_V}{192\pi^4}\left(\frac{m_{A'}}{m_{\pi_D}}\right)^2\left(\frac{m_{V_D}}{m_{\pi_D}}\right)^2\left(\frac{m_{\pi_D}}{f_{\pi_D}}\right)^4 m_{A'}\beta(x,y)^{3/2} \quad T_V = \begin{cases} 3/4 & \rho_D \\ 3/2 & \phi_D \\ 18 & \text{total} \end{cases}$$

$$\Gamma(V_D \to \ell^+\ell^-) = \frac{16\pi\alpha_D\alpha\epsilon^2 f_{\pi_D}^2}{3m_{V_D}^2}\left(\frac{m_{V_D}^2}{m_{A'}^2 - m_{V_D}^2}\right)^2\sqrt{1-\frac{4m_\ell^2}{m_{V_D}^2}}\left(1+\frac{2m_\ell^2}{m_{V_D}^2}\right)m_{V_D} \times \begin{cases} 2 & \rho_D \\ 1 & \phi_D \end{cases}.$$

Neutral V_D to visible decay rates

- Evaluate Efficiency Vertex integral and calculate expected A' signal rate for both dark vector mesons (**ρ and φ**) independently
- Total Expected Signal is sum of expected signal for both ρ and φ
- **Make reach estimate conservative by using 1 sigma downward fluctuation of F(z) in Eff_{vtx} integral, and integrating from Z_{cut} instead of target**

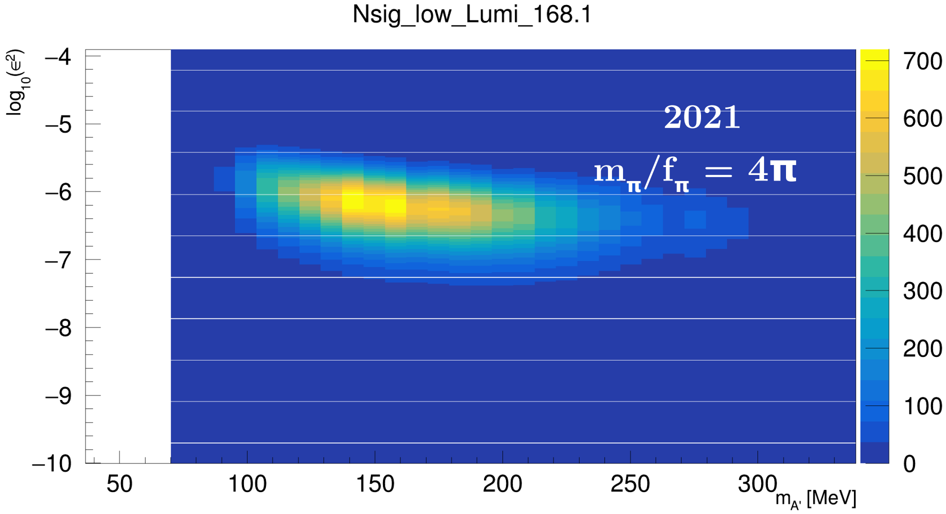
Radiative Fraction **Reco Background Rate** **Efficiency Vertex**

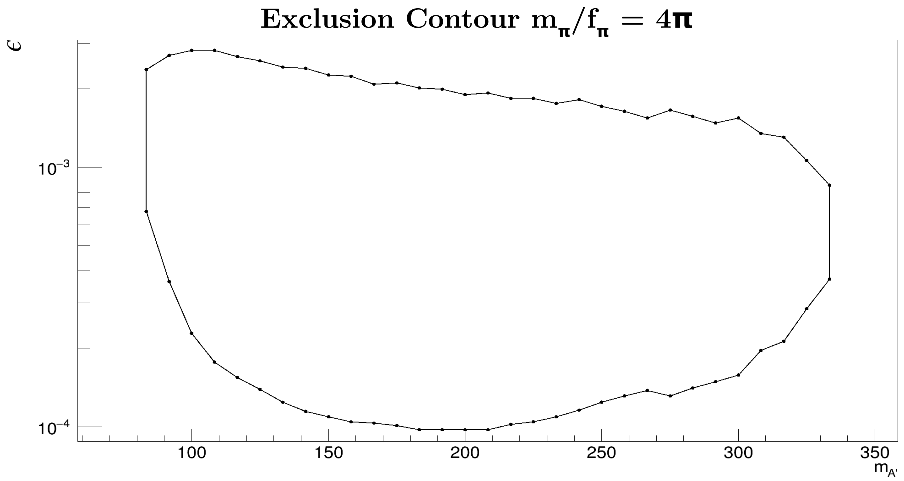
$$N_{A'sig} = \frac{3\pi m_{A'}\epsilon^2}{2N_{eff}\alpha}\frac{f_{rad}}{\zeta}\frac{dN_{CR}}{dm_{reco}}\Theta\int_{z_{tar}}^{\infty}\frac{exp^{\left(\frac{z_{target}-z}{\gamma c\tau}\right)}}{\gamma c\tau}F(z)dz$$

Total Rad Acceptance **BR(A'→$V_D\pi_D$) and BR(V_D → e^+e^-)**

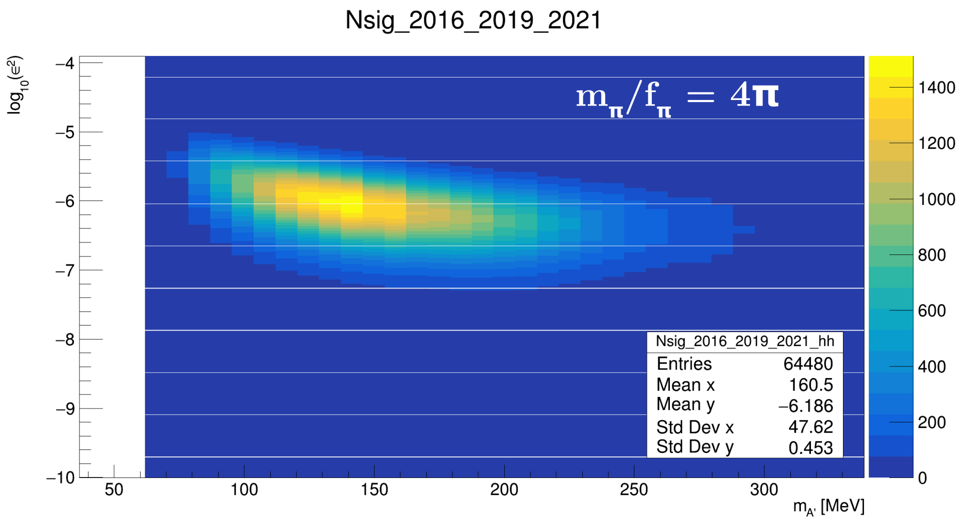
Exclusion Contour Method

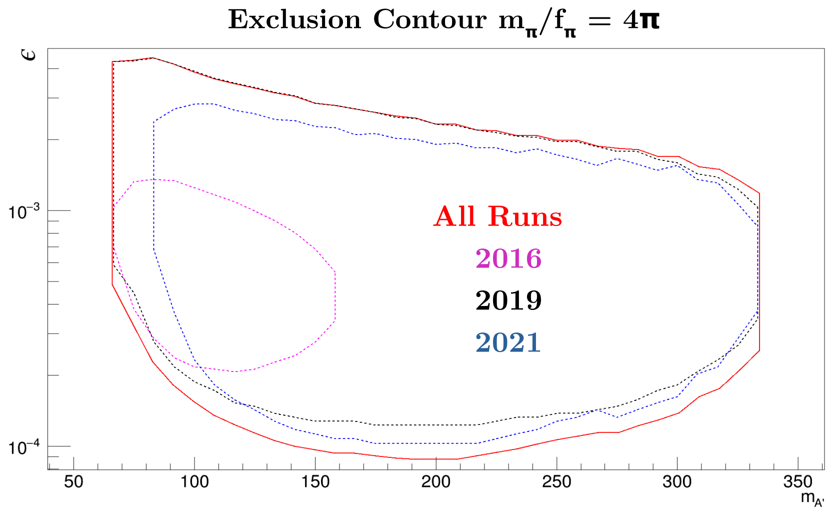
- Top plot shows HPS 2021 Expected Signal Rates using conservative method
- Make exclusion contour by finding which values of ε, for each A' mass, the Total Expected Signal (Nsig) goes above and then below 2.3 events
- Bottom plot shows Exclusion Contour for 2021 Lumi for the benchmark case where $\mathbf{m_\pi/f_\pi = 4\pi}$
- The **Combined** reach estimate for the three HPS runs 2016, 2019, and 2021 adds the Total Expected Signal for each run at each value of ε and A' mass
- The **Combined** Exclusion Contour is formed by finding which values of ε, for each A' mass, $\mathbf{N_{sig\_2016} + N_{sig\_2019} + N_{sig\_2021}}$ goes above, and then below, 2.3 events…
- Expected Signal and combined contours shown next slide...

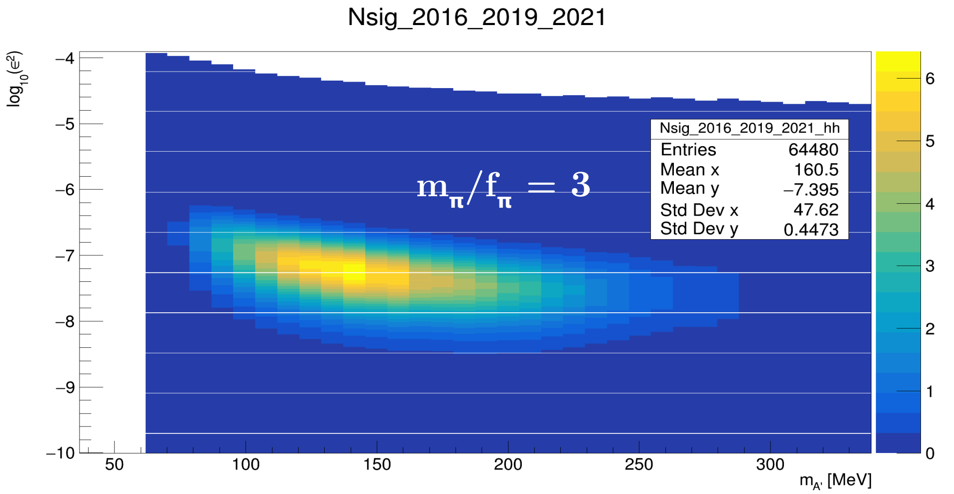


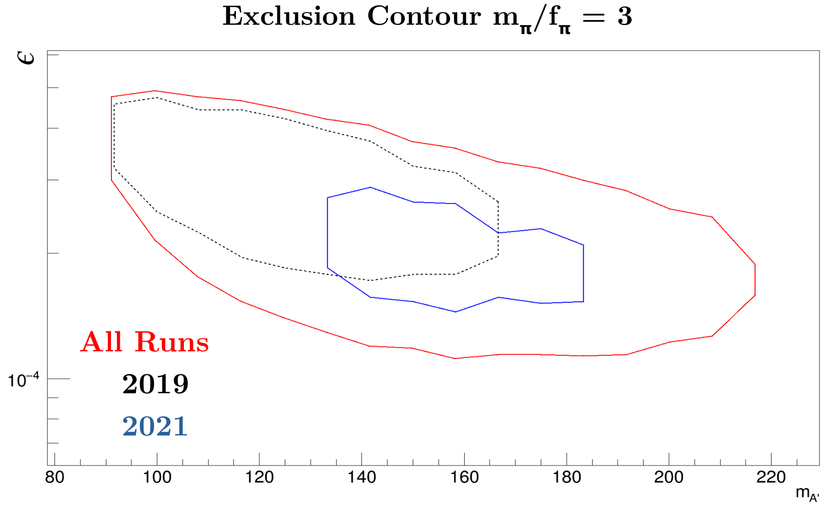


Combined SIMP Reach Estimate



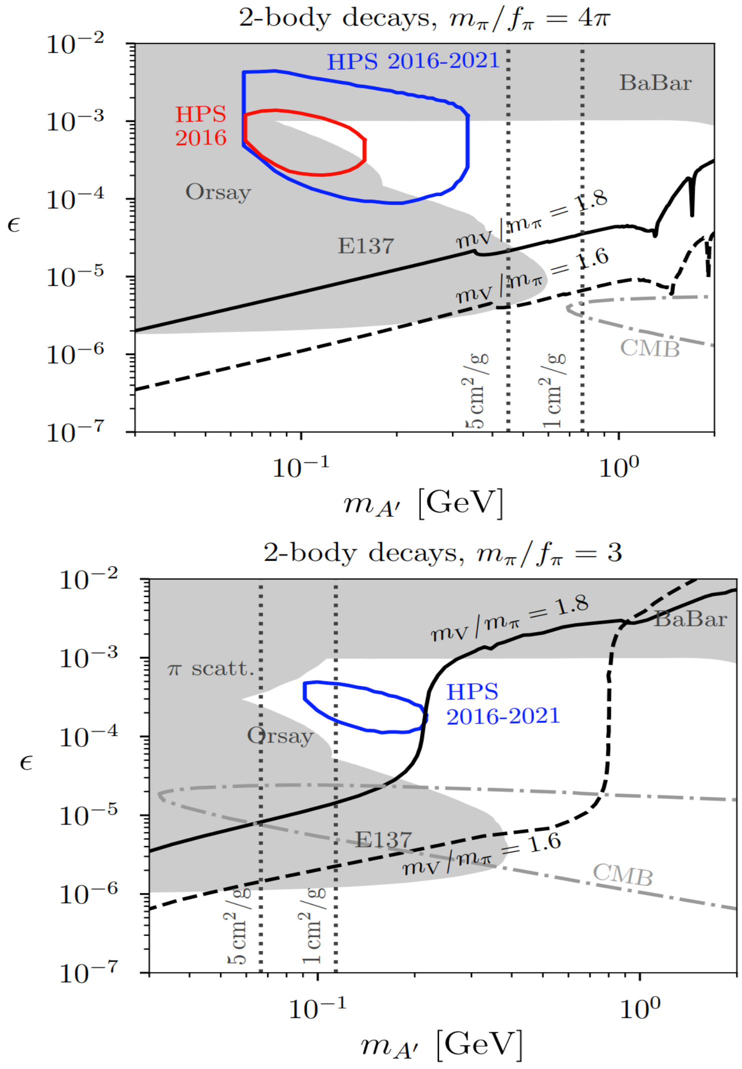






Combined Reach Estimate Final Plots

- Plotting code used in ref [1]
- HPS sensitivity plotted using combined reach estimate contour values for the two benchmark cases
- Plots show existing constraints and HPS sensitivity to signals of strongly interacting hidden sectors where only hidden sector vector mesons (V_D) that mix with A' can decay to SM particles via 2-body V_D decay
- Shaded regions represent existing exclusions
  - Orsay and E137 are beam dumps sensitive to long-lived decays to SM
  - BaBar sets upper limits on A' coupling to e^+e^-
- Hidden sector pions make up all of dark matter along solid(dashed) black lines for m_V/m_π =1.8(1.6), while dm is overabundant below lines
- Red HPS 2016 contour shows sensitivity for 2016 SIMP analysis



SIMPs Reach Estimate Conclusion

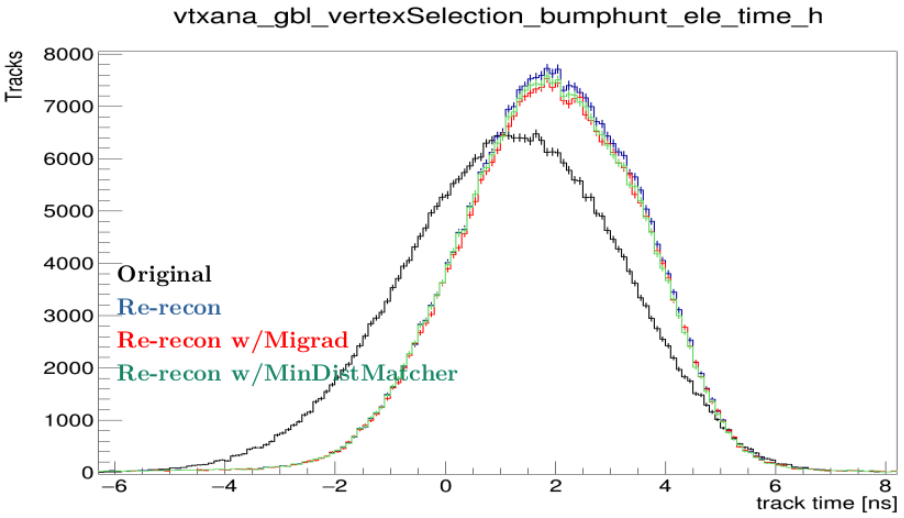
- Combined SIMPs Reach Estimate for runs 2016, 2019, 2021 is completed and final reach plots exist
- Will work on writing documentation for reach estimate process and plotting code
- Intend to request official approval of reach plots

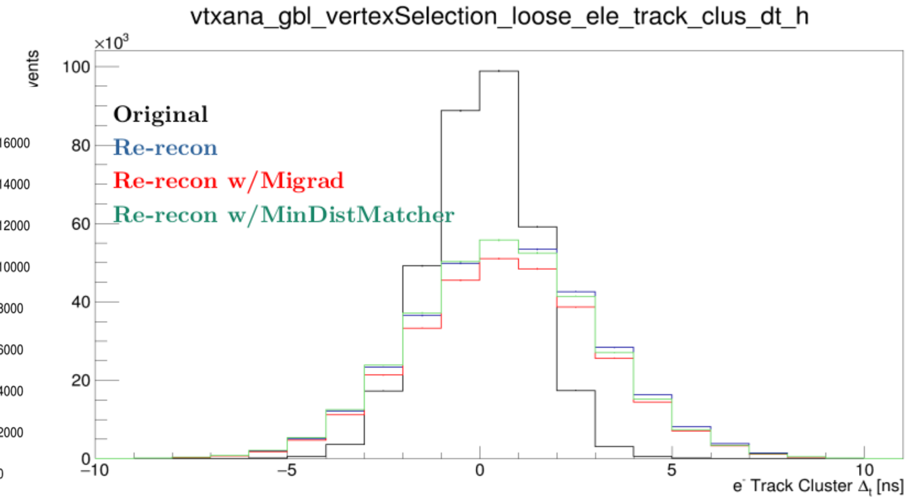
Track Time Changes
Hps-java-4.2 and Hps-java-5.1

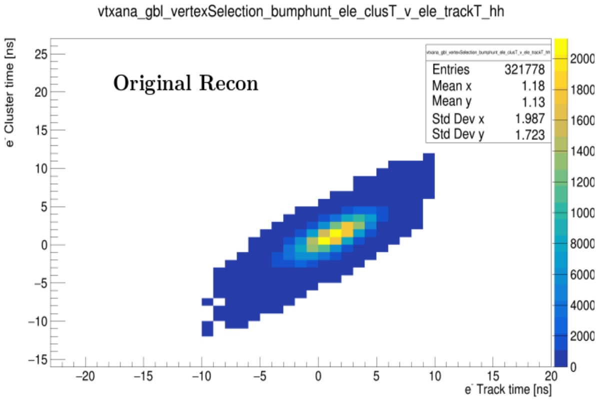
Introduction

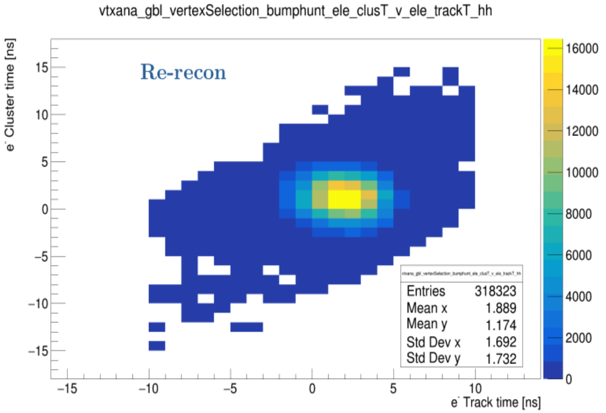
- Previously showed when re-reconstructing pass4 data with standard SeedTracker_GBL recon steering-file, Track time resolution is improved, but correlation between Track and matched Ecal Cluster times gets worse
  - Selected vertex Track-Cluster dt is wider than expected
  - Re-reco track time biased
- Original pass4 data recon used hps-java 4.2
- Current re-recon uses hps-java 5.1
- Want to figure out what’s causing change in track time between hps-java versions, and see if we can recover the correlation between track time and matched cluster time
- Ran re-reconstruction on one file from pass4 data for different cases
  - **Hps-java-4.2_GBL** (standard PhysicsRun2016FullRecon.lcism using SeedTracker+GBL)
  - **Hps-java-5.1_GBL** (standard PhysicsRun2016FullRecon.lcism using SeedTracker+GBL)
  - **Hps-java-5.1_GBL_noClusterRFTimeCorr** (removed ClusterRFTimeCorrectionDriver from steering-file)
  - **Hps-java-5.1_GBL_rm_cams_subTT** (removed subtractTriggerTime code added by Cam)
  - **Hps-java-5.1_KF** (standard PhysicsRun2016FullRecon.lcsim using KalmanFullTracks)
  - **Hps-java-5.1_KF_noClusterRFTimeCorr** (removed ClusterRFTimeCorrectionDriver from steering-file)
  - **Hps-java-5.1_KF_rm_cams_subTT** (removed subtractTriggerTime code added by Cam)
  - **Hps-java-5.1_KF_MinDistanceMatcher** (using Minimum Distance Track-Cluster matching algo)

- Top right plot compares selected vertex electron track times
  - “original” refers to hps-java-4.2 reco
  - Improved track time resolution, shift in bias
- Bottom right plot shows matched Track-Cluster time residuals
  - Re-recon cases have wider dt
- Bottom left two plots show track-time v cluster-time
  - Hps-java-4.2 recon (“original”) shows linear correlation between matched Track and Cluster times
  - Hps-java-5.1 recon de-correlated
- What changed between versions 4.2 and 5.1 that cause this?

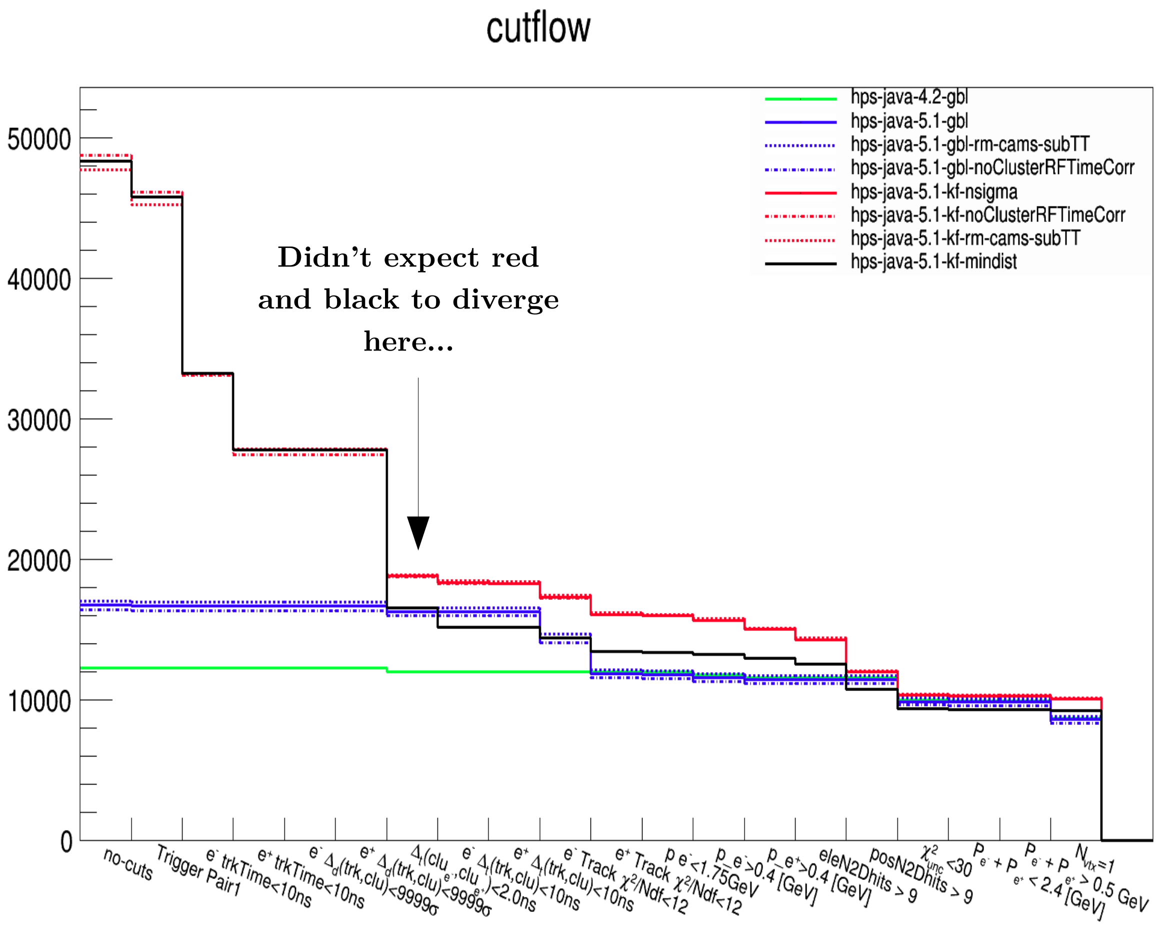


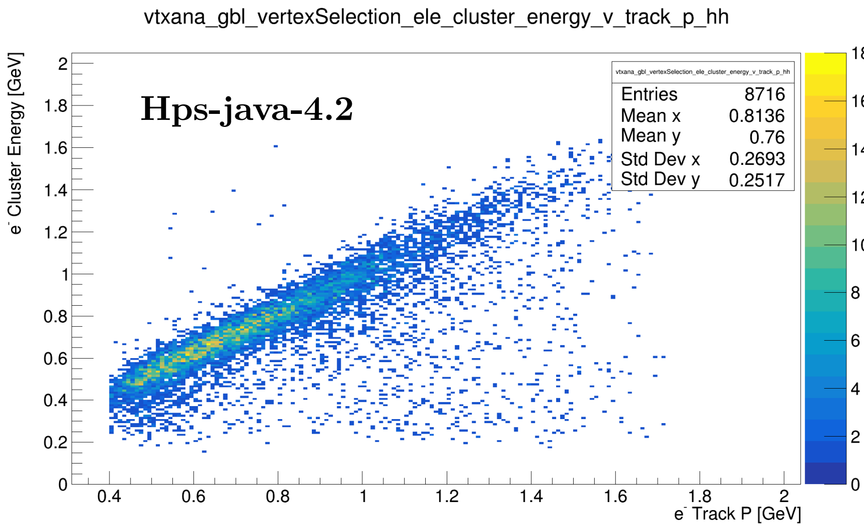


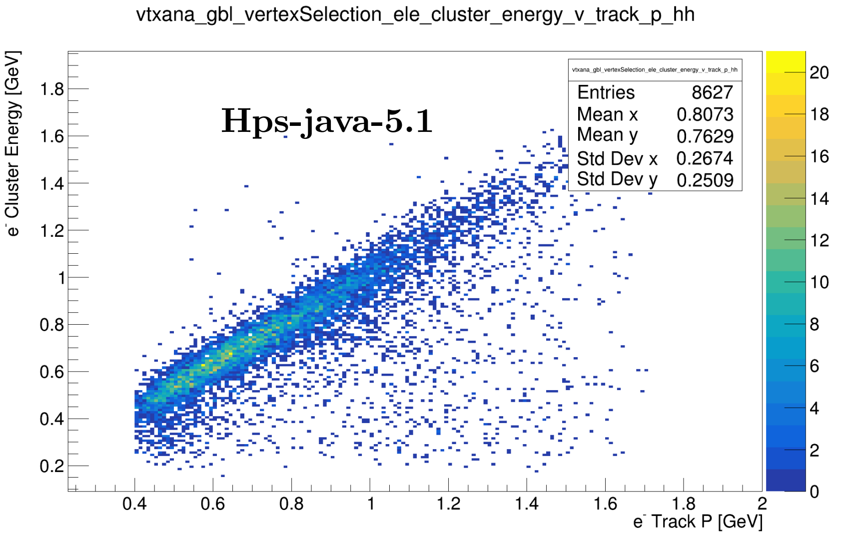


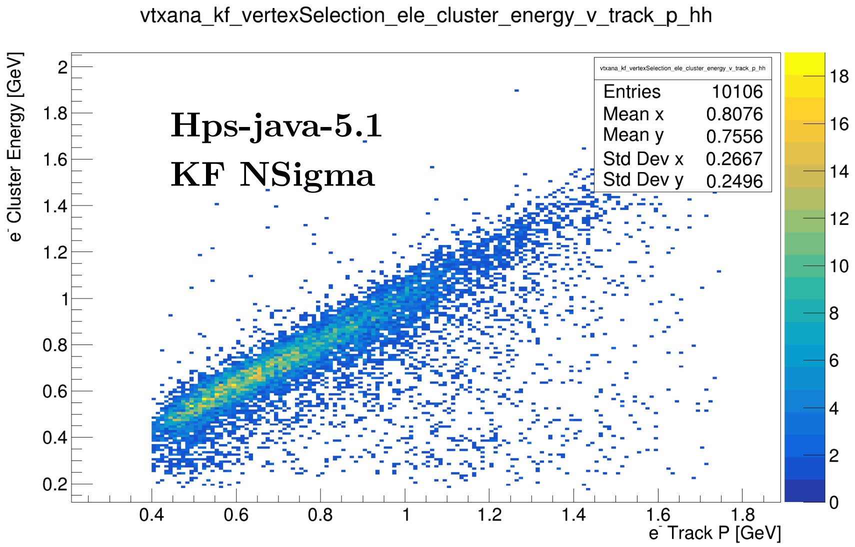


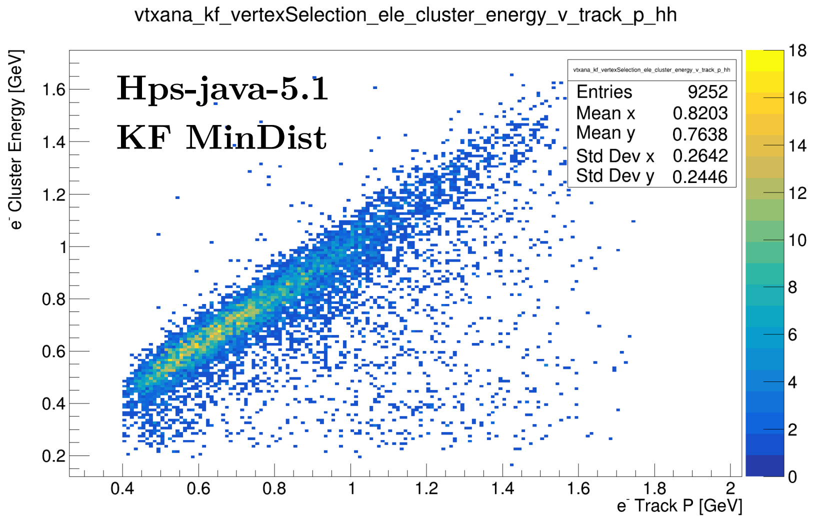
- Reconstructed one file from run 7800 for the various cases discussed in intro
- Plot shows vertex selection cutflow used to compare
- Not clear why the ele/pos cluster time difference cut cuts more vertices for KF-MinDistance than KF-Nsigma…
- Next slides look at track and cluster times



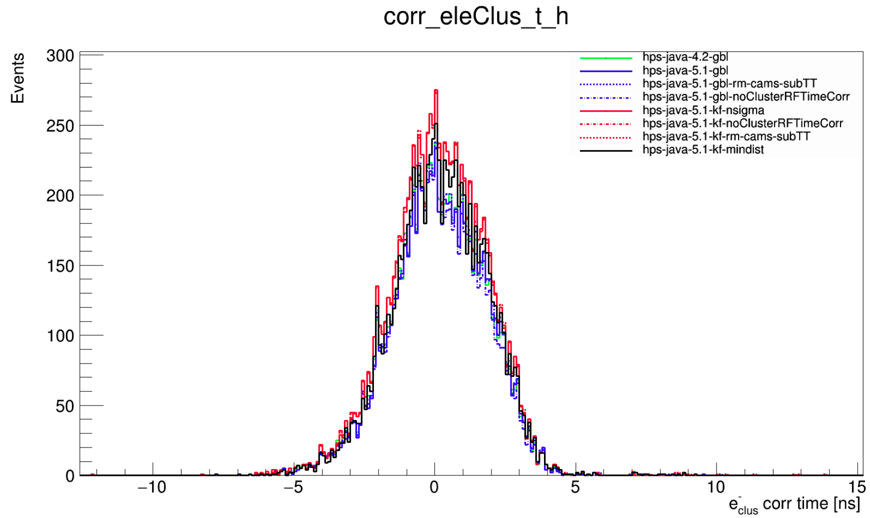

vtxana_gbl_vertexSelection_ele_cluster_energy_v_track_p_hh
Hps-java-4.2
Entries 8716
Mean x 0.8136
Mean y 0.76
Std Dev x 0.2693
Std Dev y 0.2517
e- Cluster Energy [GeV]
e- Track P [GeV]

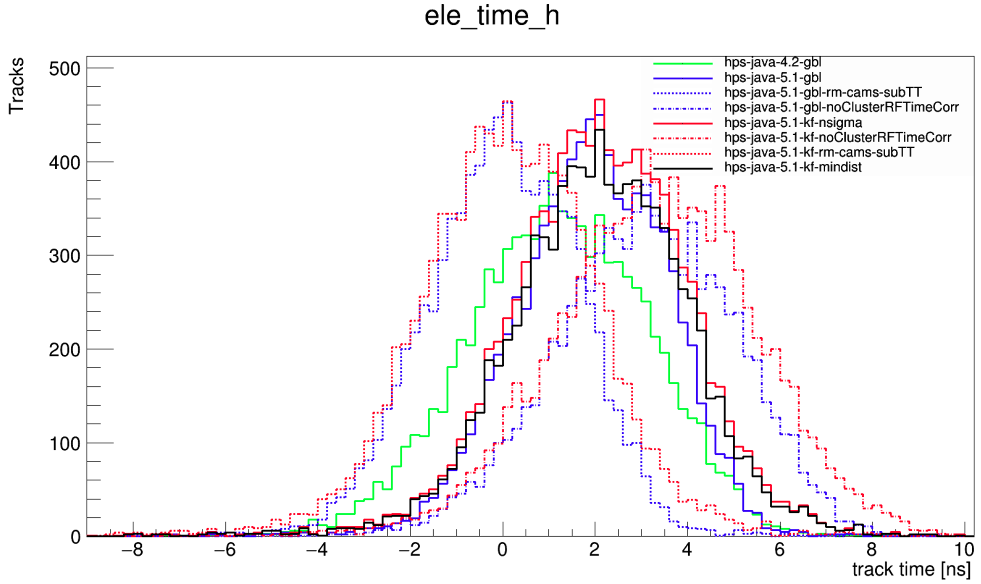

vtxana_gbl_vertexSelection_ele_cluster_energy_v_track_p_hh
Hps-java-5.1
Entries 8627
Mean x 0.8073
Mean y 0.7629
Std Dev x 0.2674
Std Dev y 0.2509
e- Cluster Energy [GeV]
e- Track P [GeV]

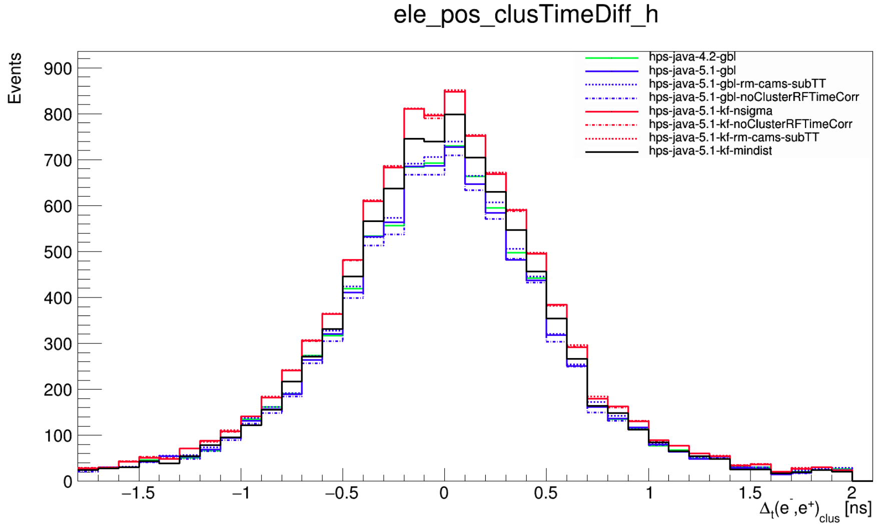

vtxana_kf_vertexSelection_ele_cluster_energy_v_track_p_hh
Hps-java-5.1
KF NSigma
Entries 10106
Mean x 0.8076
Mean y 0.7556
Std Dev x 0.2667
Std Dev y 0.2496
e- Cluster Energy [GeV]
e- Track P [GeV]

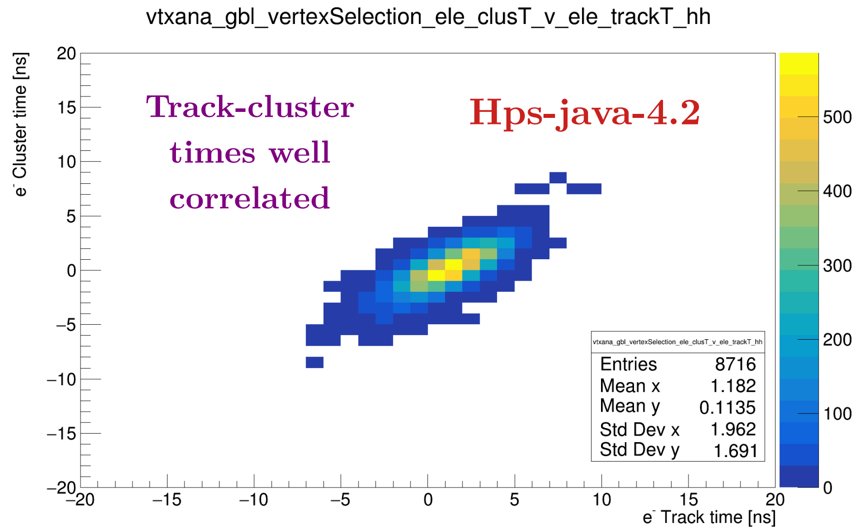

vtxana_kf_vertexSelection_ele_cluster_energy_v_track_p_hh
Hps-java-5.1
KF MinDist
Entries 9252
Mean x 0.8203
Mean y 0.7638
Std Dev x 0.2642
Std Dev y 0.2446
e- Cluster Energy [GeV]
e- Track P [GeV]

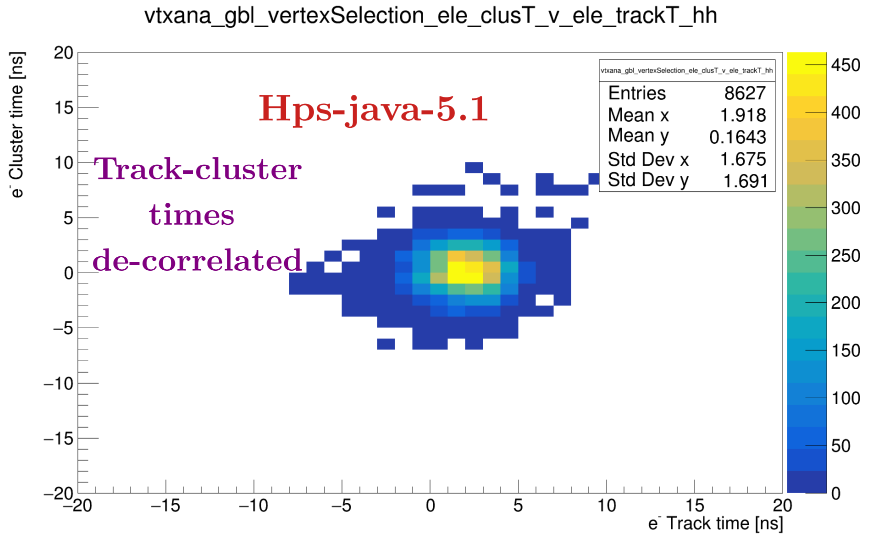
- (Top right) Cluster time distribution not different between reconstruction configurations
- (Bottom left) Track times do change…
- The ClusterRFTimeCorrectionDriver clearly improves the track time resolution
- The ‘subtractTriggerTime’ code added between hps-java versions 4.2 and 5.1 appears to just displace the track time distribution, shifting the bias from ~0ns to ~2ns
- Comparing the solid green line (v4.2) to the solid blue line (v5.1) shows a large improvement in track time resolution



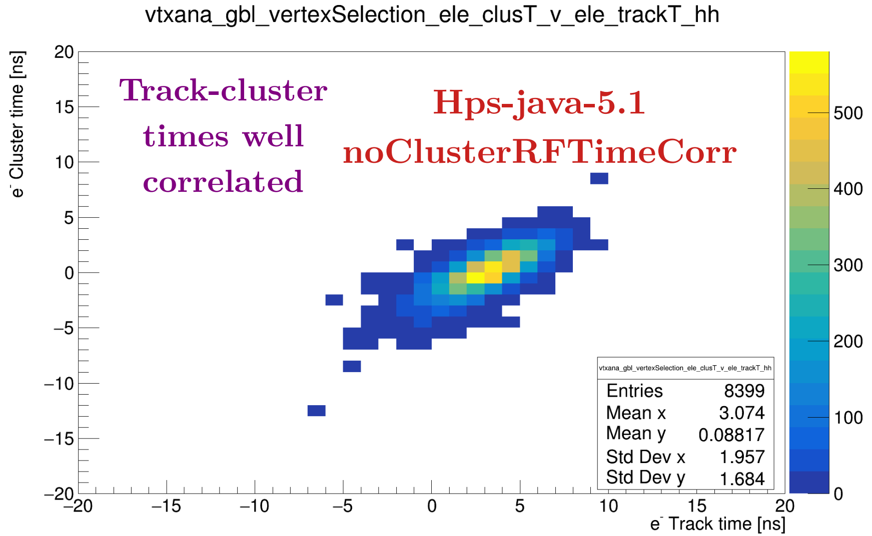


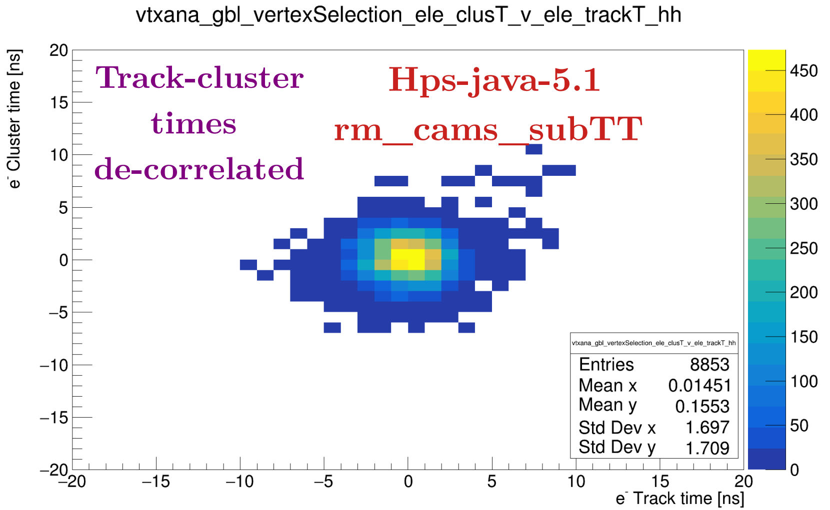




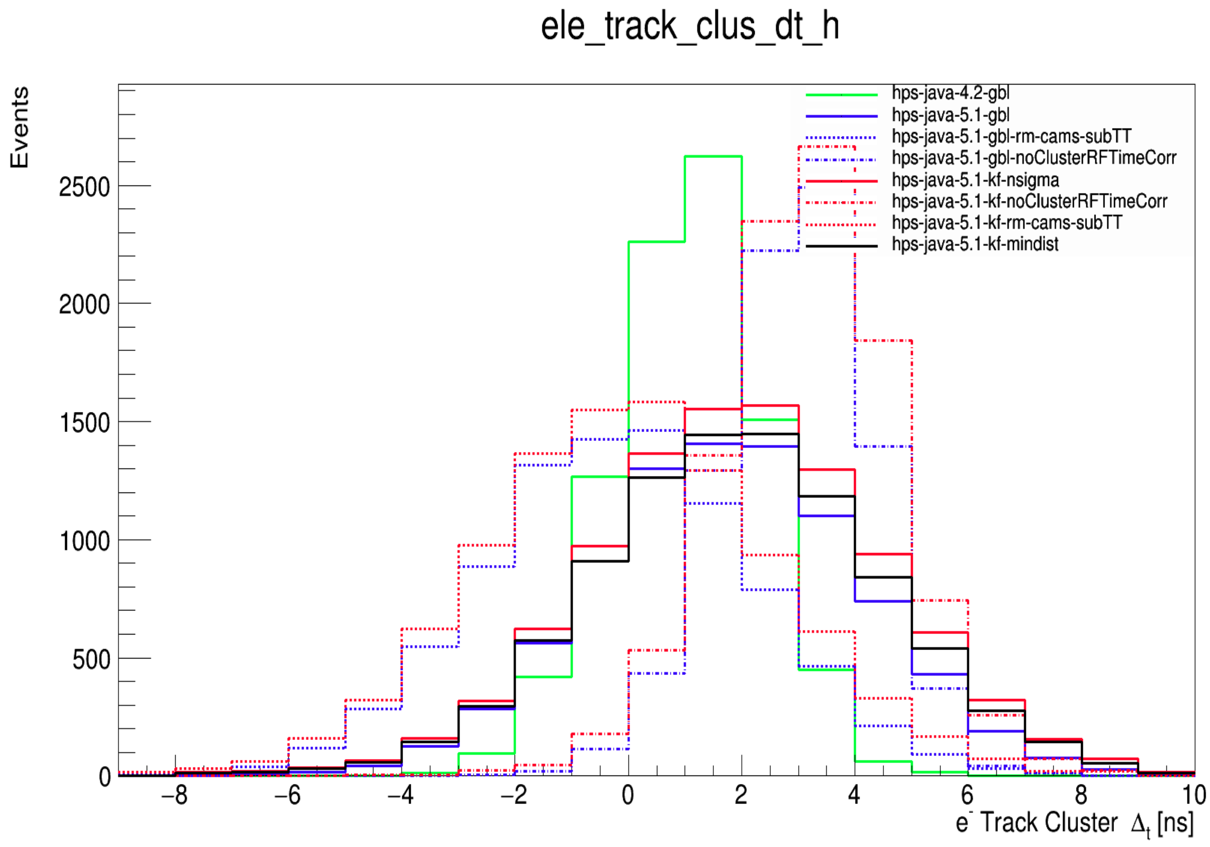


All GBL Tracks





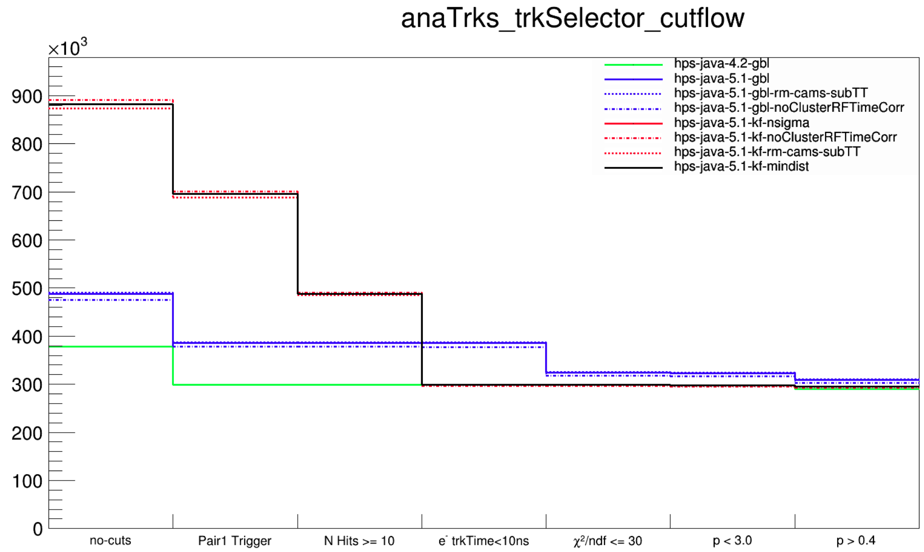
- Plot shows selected vertex matched Track-Cluster pair time residuals
- Hps-java-4.2, and hps-java-5.1 where the ClusterRFTimeCorrectionDriver is disabled (GBL and KF), both have more peaked time residuals
- Other cases show much wider distribution in track-cluster time residuals
- We've seen that the ClusterRFTimeCorrectionDriver significantly improves the track time resolution, but it is de-correlating the track-cluster pair times...

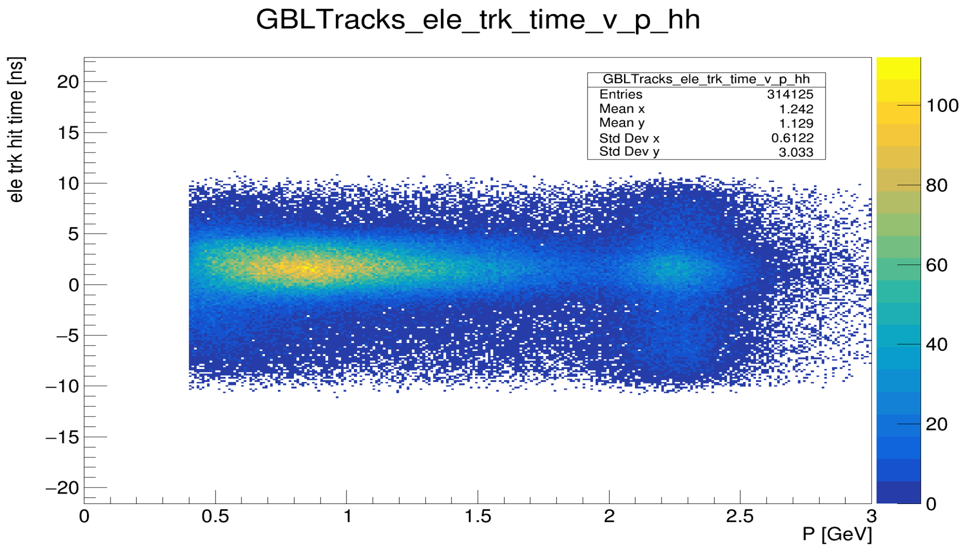


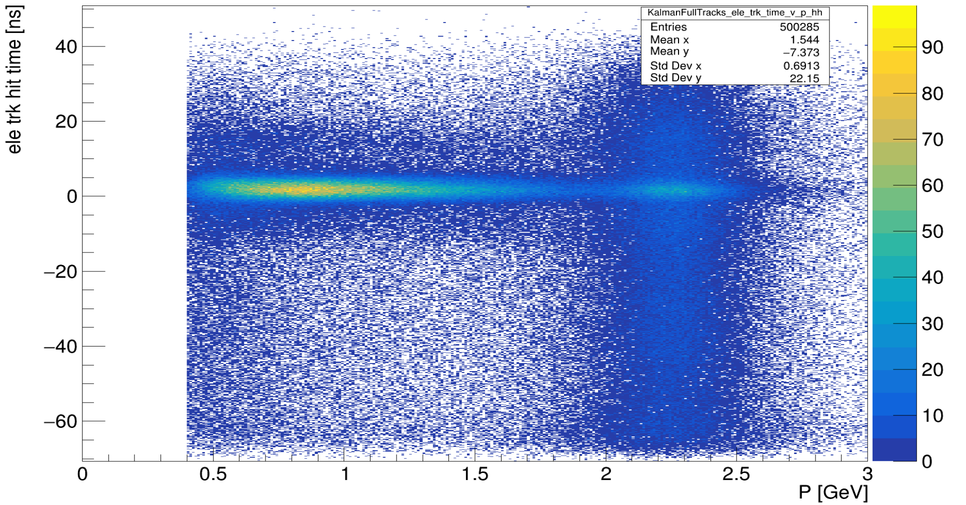
ClusterRFTimeCorrDriver

- Driver chooses highest energy cluster within trigger time window in an event and uses the RF time to set the trigger time of the event
- Start by getting the "ideal" trigger time window for run from conditions db
- Read in event ecal clusters, choose highest energy cluster in trigger time window
- Select highest seed energy cluster in time window
- Read in rf time using RFHits
- Relationship between rf-time and selected cluster time defined as "jitter", used to modify selected cluster time
- "TriggerTime" collection is created which relates jitter-modified selected cluster time and the cluster seeds
- TriggerTime collection is used in RawTrackerHitFitterDriver under "subtractRFTime" setting
- RF time jitter is read in from collection and applied to the hit fits T0 parameter for the event
- ...maybe these jitter corrections are only being applied to tracker hits, but should also be applied to the clusters…?

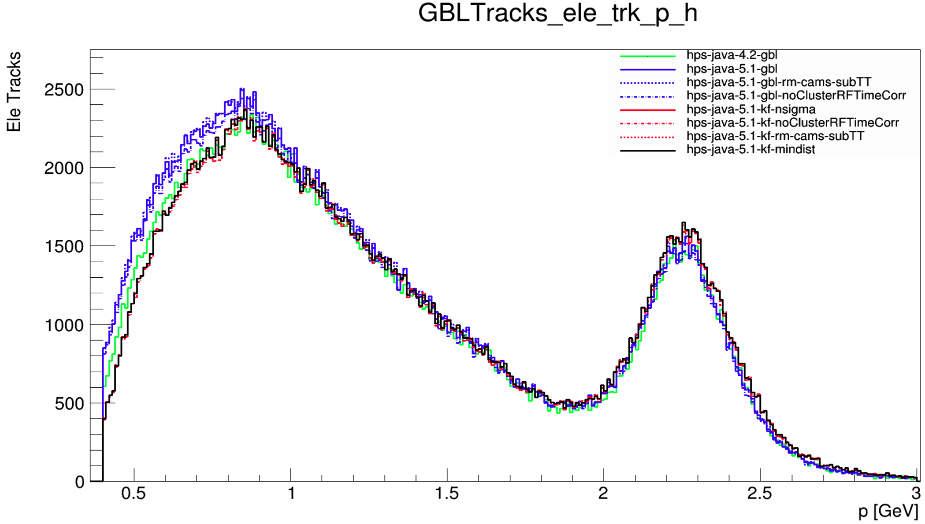
- The following plots are at track level (not tracks on vertex)
- Plots show electron track times v momentum for GBL (Top) and KF (Bot Right) with no cuts
- Bottom left plot shows track selection cutflow…
  - Built-in +-10ns cut on GBL Tracks must be included for KF Tracks
  - Turns out this is why KF Tracks show a larger FEE peak in past studies…
  - Disappears when cutting on time...whoops…

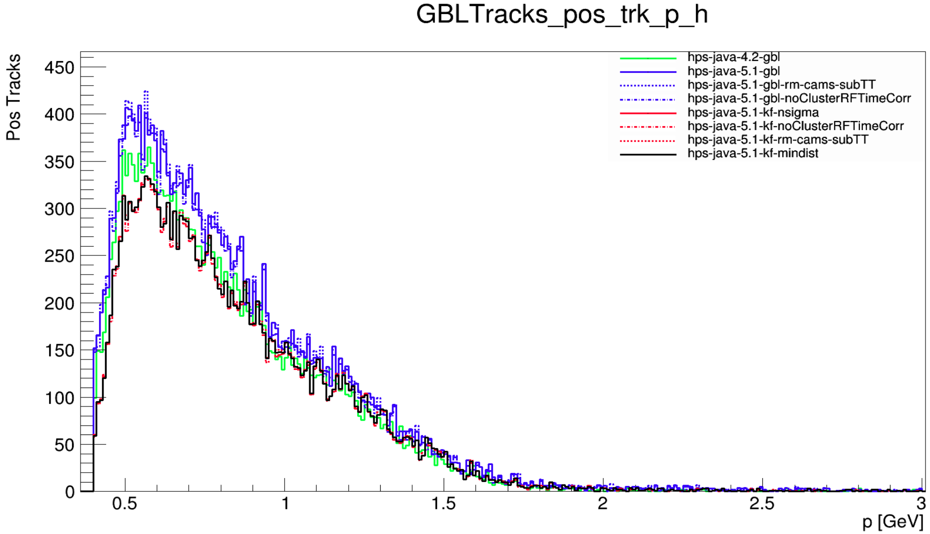




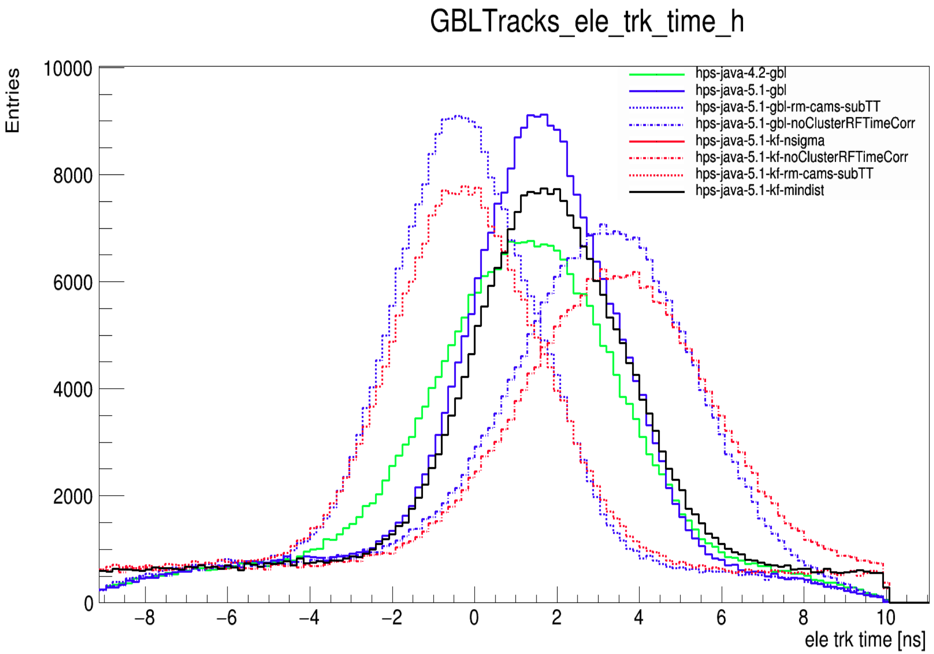


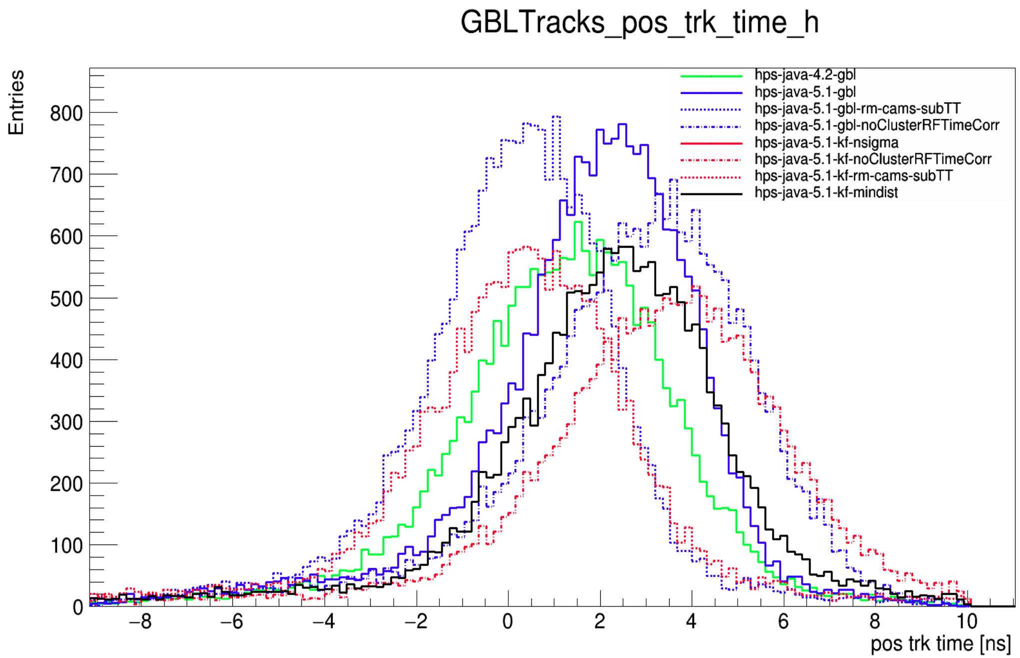
- Plots show electron (Top) and positron (Bot) track momentum
- Past studies have shown high fake rate for low momentum GBL tracks, so not surprising to see less KF tracks in that region
- Note that the FEE peaks are close for GBL and KF, whereas I've previously shown more KF Fee's…turns out that was due to baked in GBL track time cut…again whoops…



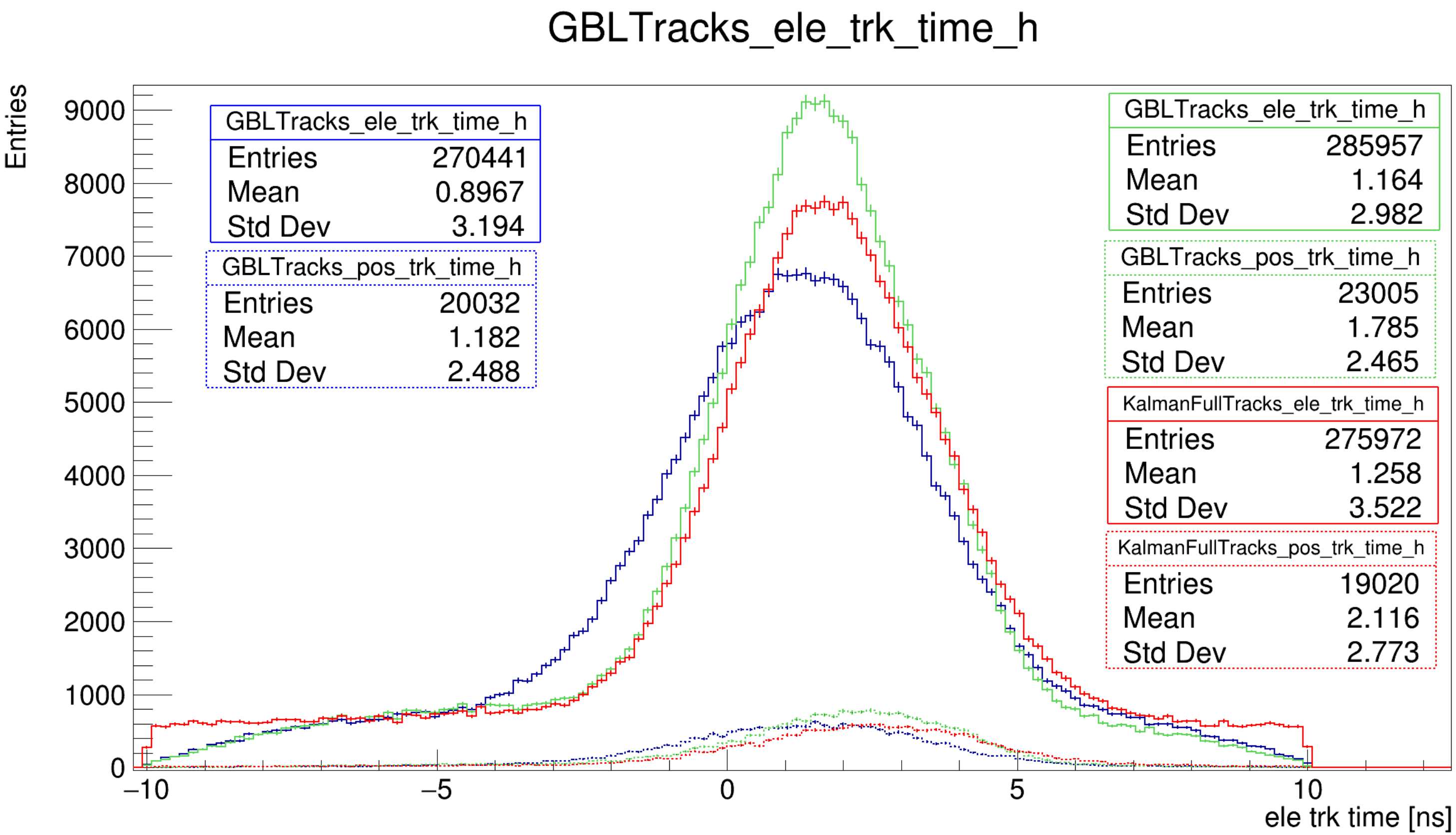


- Plots show electron and positron track times
- Shows essentially same thing as the vertex level track times

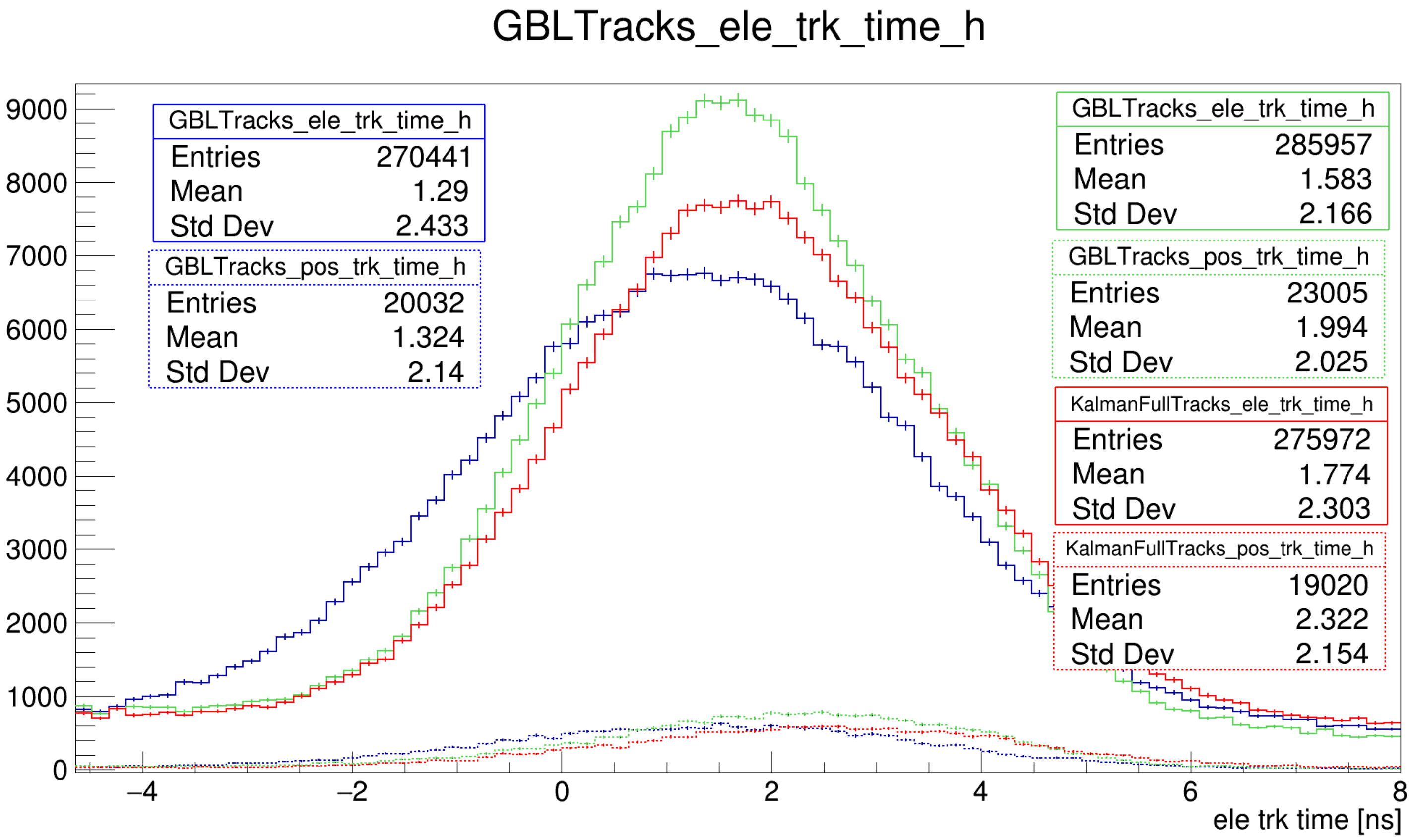




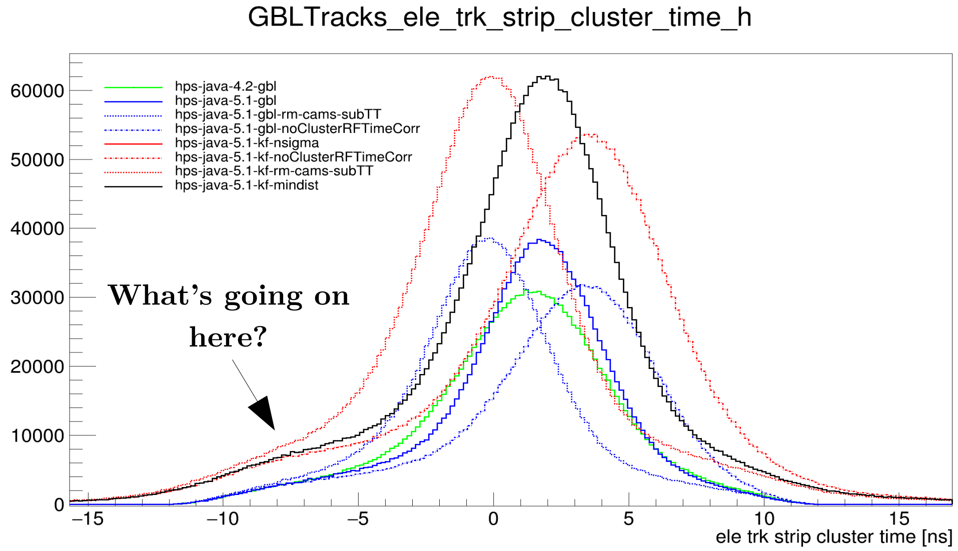
- Hps-java-4.2 GBL Tracks (Nsigma Matcher)
- Hps-java-5.1 GBL Tracks (Nsigma Matcher)
- Hps-java-5.1 KF Tracks (Nsigma Matcher)
- Next slide zooms into Gaussian core to better compare resolutions

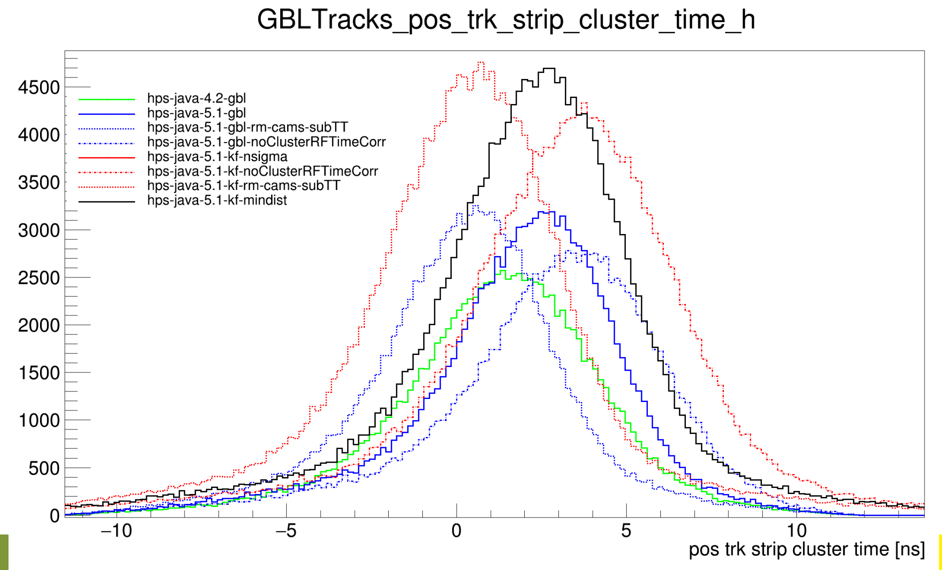


- Hps-java-4.2 GBL Tracks (Nsigma Matcher)
- Hps-java-5.1 GBL Tracks (Nsigma Matcher)
- Hps-java-5.1 KF Tracks (Nsigma Matcher)
- Electron track time resolution improves between v4.2 and v5.1
- KF time resolution slightly worse than GBL
- Positron track time resolution less improved (if at all)

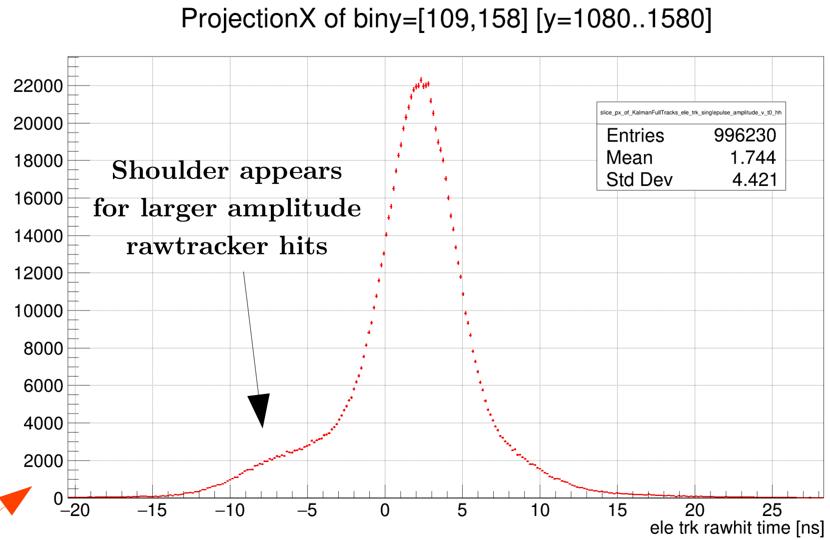


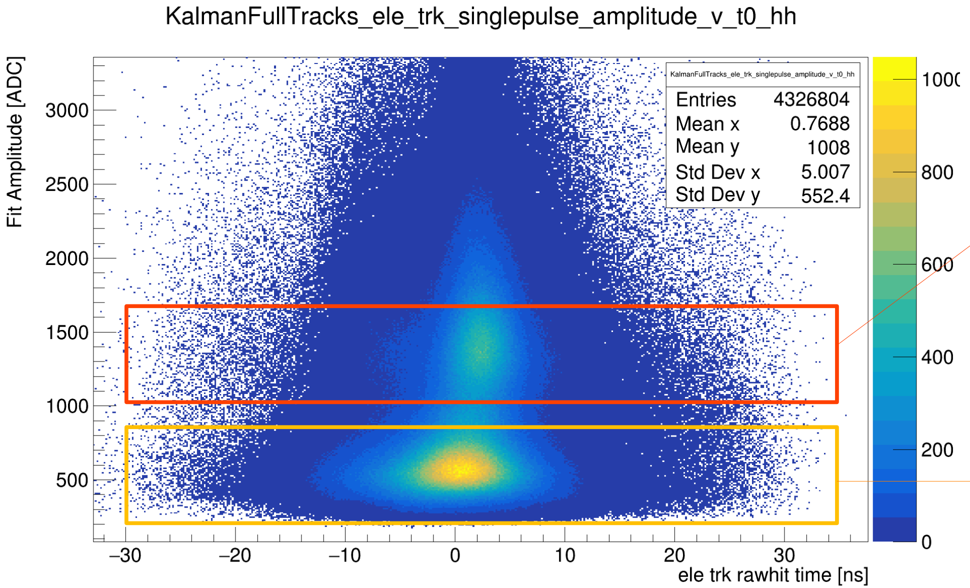
- Plots show strip clusters on tracks
- Note that KF tracks have twice as many strip clusters as GBL
- Differences between each case are same as already discussed…
- Top plot shows electron strip cluster times
  - Shoulder on left side of track time distribution present in all cases...

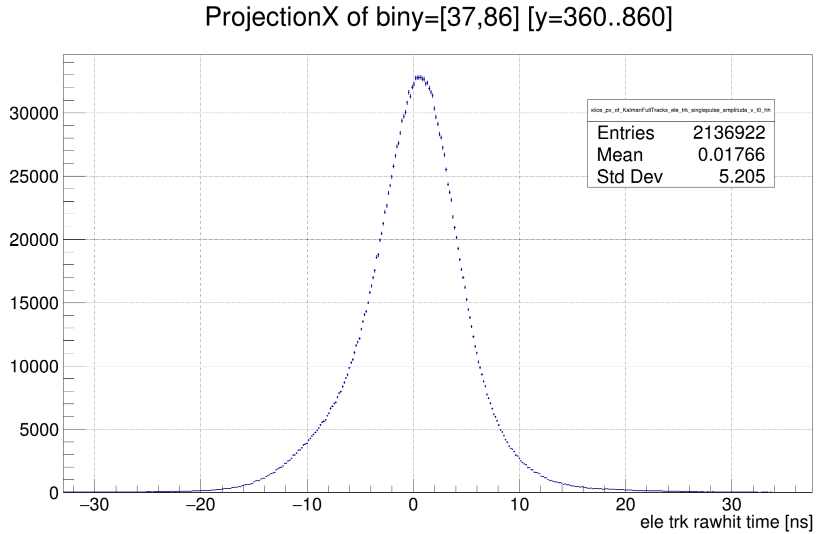




- Bottom plot shows track rawhits amplitude vs t0 for hits fit with a single pulse (hps-java-v5.1)
- Larger amplitude hits have shoulder ~ -8ns







Summary and Conclusion

- The ClusterRFTimeCorrectionDriver is crucial to improving track time resolution, but is causing a de-correlation between Track and Cluster pair times
  - Seems to involve the "jitter" corrections being applied to hit fit T0
- There's a shoulder in electron track time ~-8ns that seems to be from large amplitude strip hits
  - Can investigate further at hit level…
- The "subtractTriggerTime" code added in between v4.2 and v5.1 seems to just displace the track times, and should probably not be applied to 2016 data at least…
- There are slightly more vertices for KF tracks using Nsigma Track-Cluster matcher than the "MinDistance" matcher...could be due to the track time shift or Track-Cluster dt de-correlation...will look into this once the timing is figured out…
- Still working on understanding the track time changes
- Going to move past this stuff for now and start developing analysis tools