ALICE data flow

Data Preparation Group

Analysis Tutorial

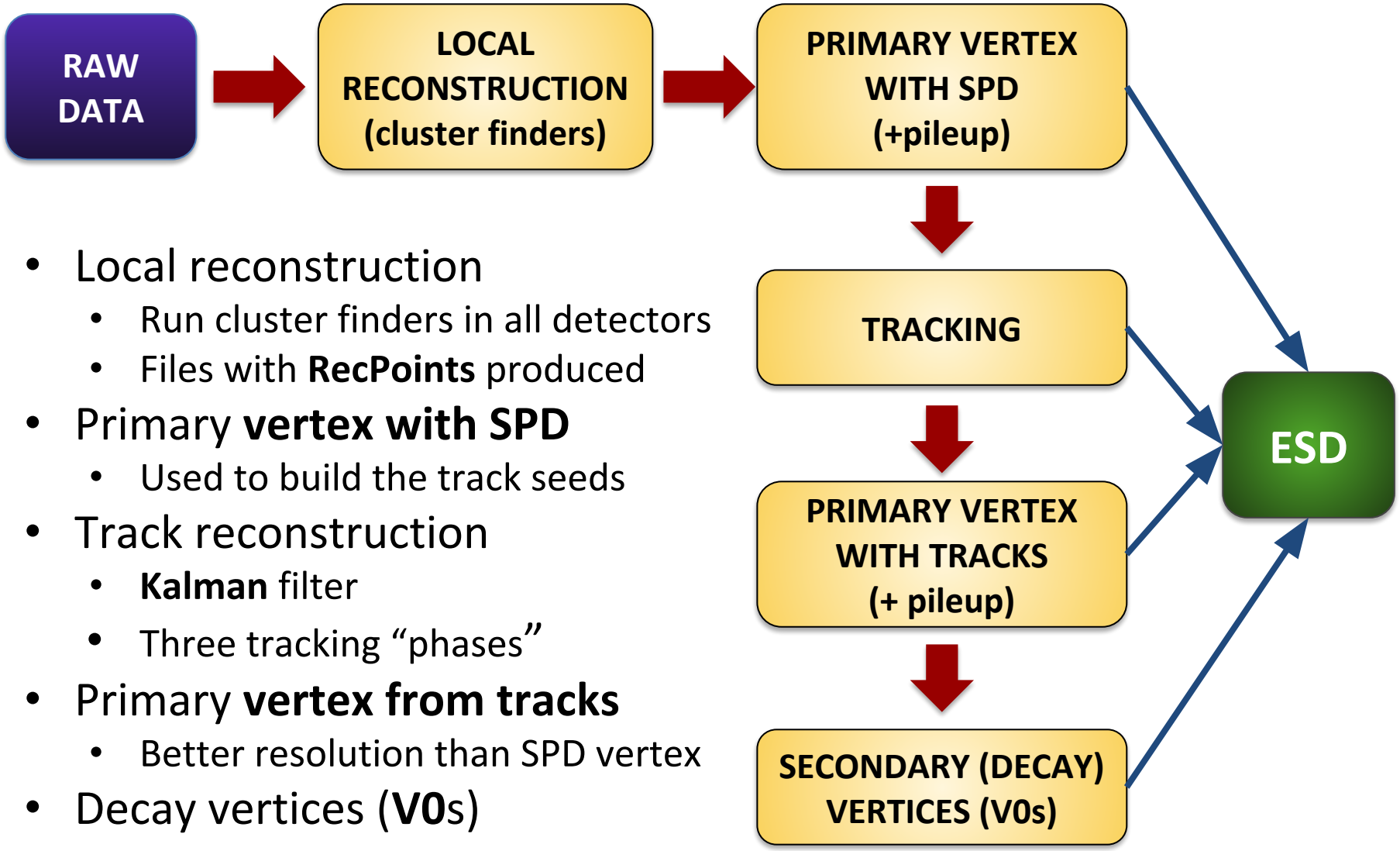
2 November 2017

Outline

- Reconstruction steps **from raw data to ESDs**
  - Track, primary vertex, decay vertices
- **Calibration and CDB**
  - Online and offline calibration, reconstruction passes
- **Monte Carlo simulation** chain
- **ESD and AOD**
  - ESD and AOD contents, ESD->AOD filtering
- **Event properties and selection**
  - Physics selection, pileup rejection, centrality determination
- **Track properties and selection**
  - Track cuts (ESD vs. AOD), TPC-ITS matching efficiency, …

RECONSTRUCTION: FROM RAW DATA TO ESD

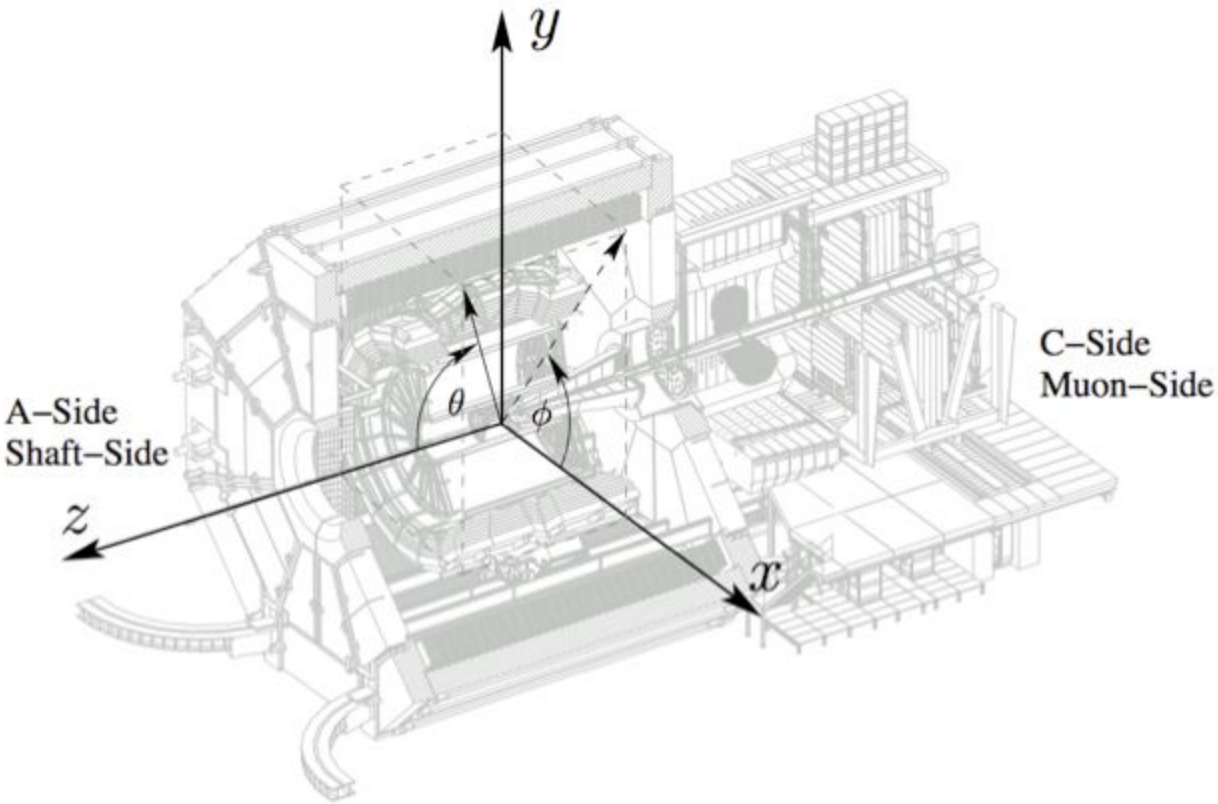
General reconstruction strategy



- Local reconstruction
  - Run cluster finders in all detectors
  - Files with **RecPoints** produced
- Primary **vertex with SPD**
  - Used to build the track seeds
- Track reconstruction
  - **Kalman** filter
  - Three tracking "phases"
- Primary **vertex from tracks**
  - Better resolution than SPD vertex
- Decay vertices (**V0**s)

ALICE global reference system

- **z axis** along the beam direction
  - Muon armin the negative z direction (negative pseudorapidity η)
- **x axis** points towards the centre of the LHC
- **y axis** points upwards

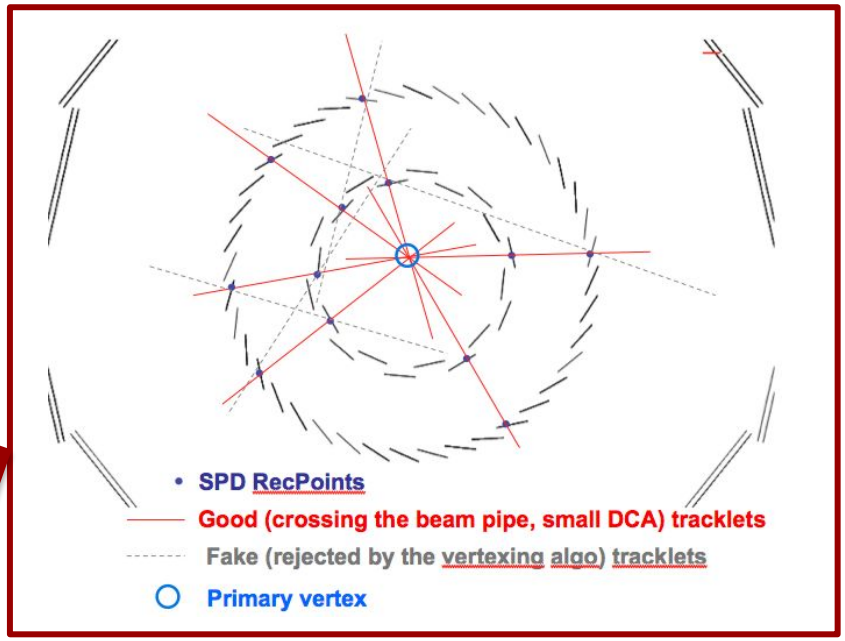


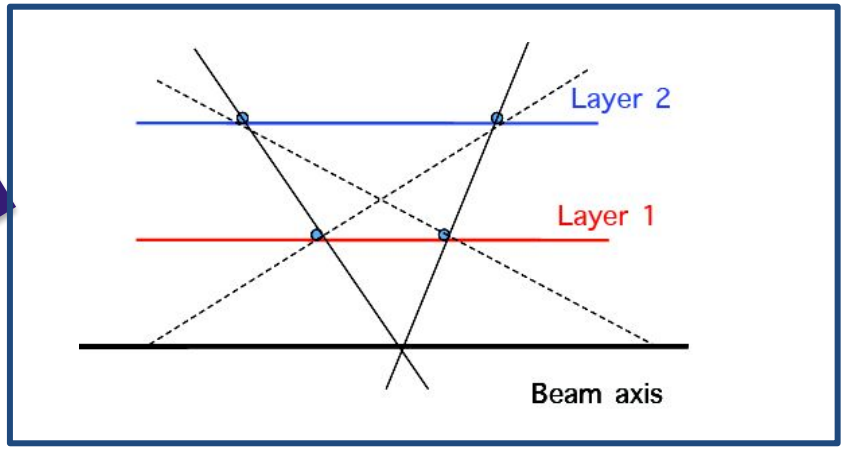
Local reconstruction

- **Local detector reconstruction: clusterization step**
  - Raw data of each detector data are converted into **clusters**
  - Clusterization is performed separately for each detector
- **Cluster** (or **RecPoint**) = groups of adjacent detector cells firing
  - Corresponds to a hit (energy deposition) produced by a ***crossing particle*** (or by a ***shower*** in the calorimeters)
  - Characterized by:
    - Positions, signal amplitudes, signal times, cluster shapes…
    - … and their associated errors
  - Used as input for track (and tracklet) reconstruction for tracking detectors (ITS, TPC, TRD)
  - Matched to tracks for PID detectors (TOF, HMPID, EMCAL...)
  - Clusters from calorimeters (EMCAL, PHOS) stored in ESDs
    - Used, e.g. for photon reconstruction: a cluster with no tracks in the vicinity is a neutral particle candidate

Primary vertex from SPD

- **Interaction vertex** position reconstructed from pairs of SPD points (tracklets)
  - Search for **pileup** from SPD tracklets not pointing to the "main" interaction vertex
- **3D reconstruction**
  - Default in pp and p-A
  - Not used in A-A where the faster Z-only vertex is used
- **Z-only reconstruction**
  - Fallback in pp and p-A if the 3D fails (low multiplicity events)
  - Default in A-A

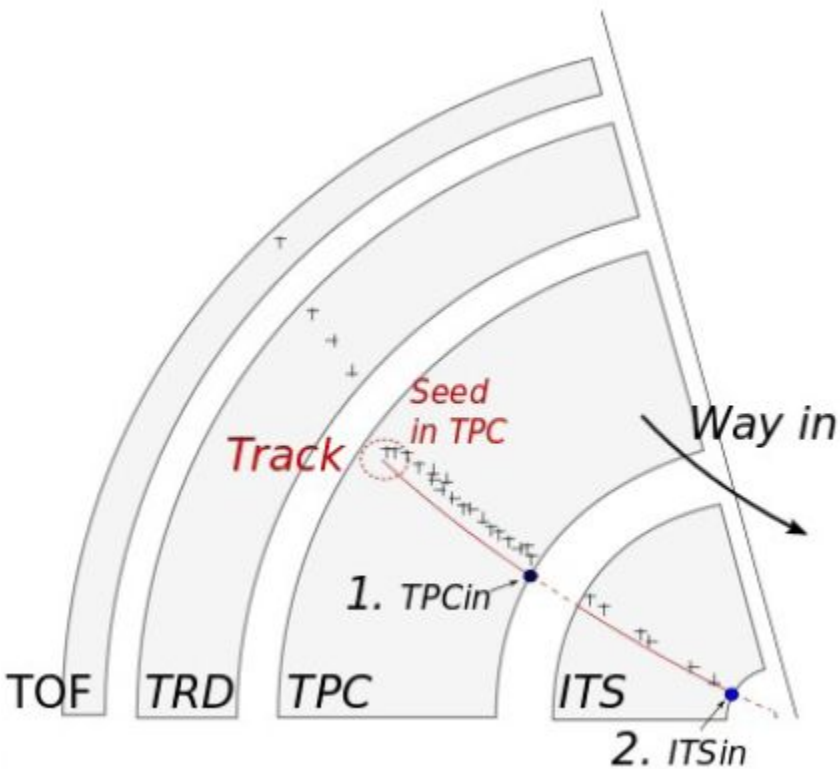




Tracking: Kalman filter

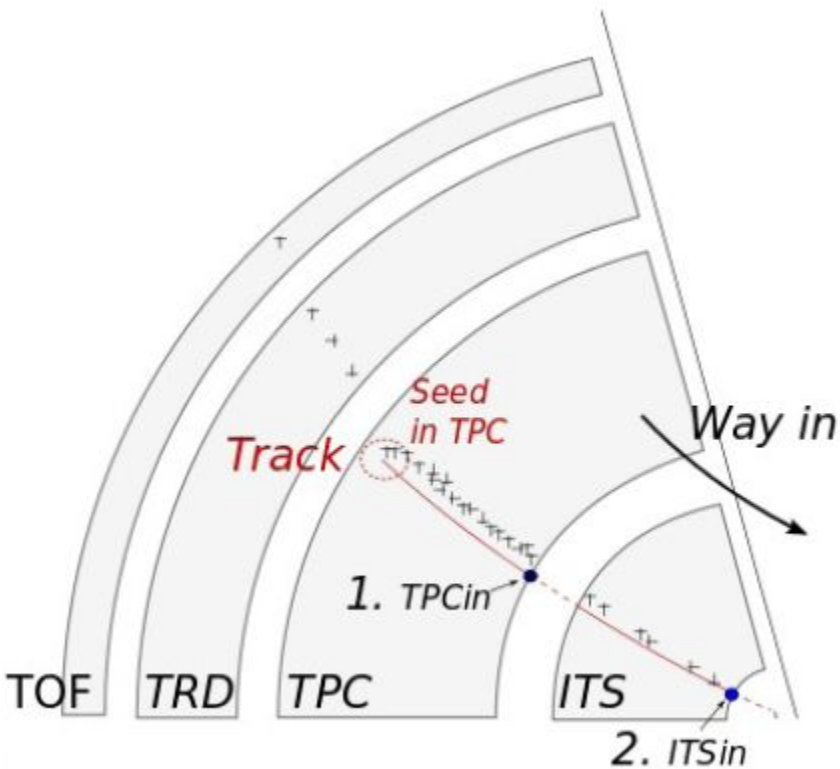
- **Local method**: the track parameters are estimated 'locally' at a given point in space
  - Naturally accounts for local track peculiarities: stochastic physics processes (multiple scattering, energy loss), B field
- Simultaneous **track recognition** (*track finding*) and **reconstruction** (*track fitting*)
  - Needs "state vector" (= track parameterisation) + "evolution equation" (= how state vector varies from current to next "step")
  - Requires to start from a track "seed" (first track candidate)
- Tracking **procedure**:
  - Extrapolate the current estimate to the next "step" (*prediction*)
  - Combine the prediction with the next state vector measurement (*filtering*) -> improved determination of track parameters
  - Repeat prediction and filtering steps as many times as we have measurements of the state vector.

Tracking: seeding



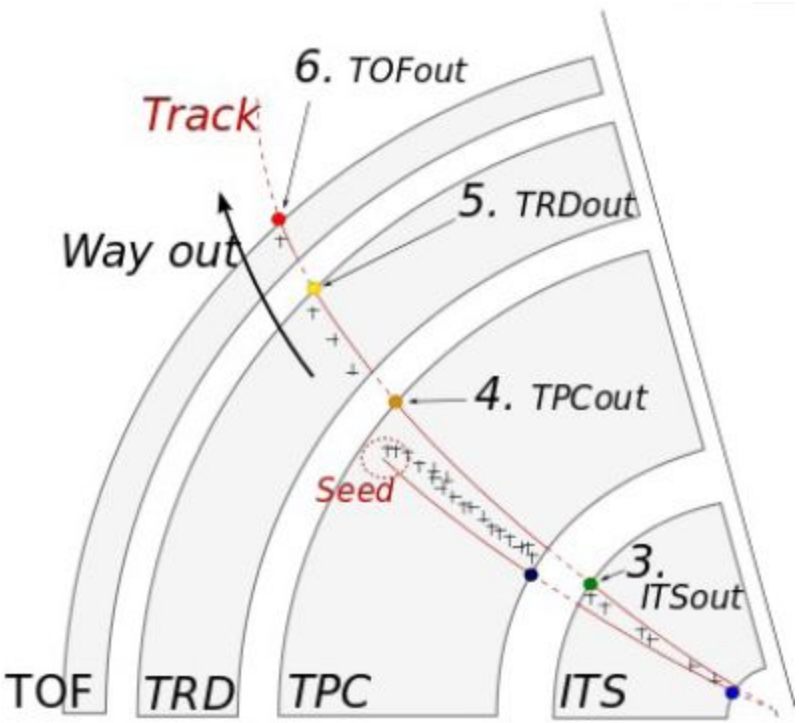
- **Track seeds** built using SPD primary vertex and pairs of TPC RecPoints in adjacent pad rows
  - Seeding starts at outer TPC radius, where hit density is minimal
  - Seeds for candidate secondary tracks built from 3 TPC points w/o using the SPD vertex

Tracking: 1st phase -> inward



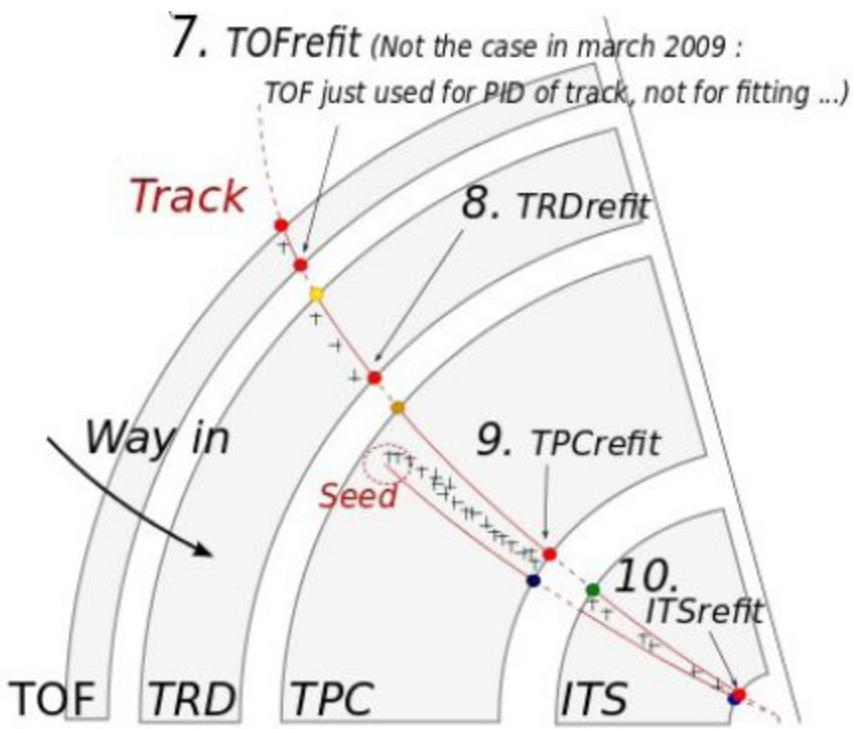
- Track seeds are **projected inward** using a Kalman filter algorithm
  - Updated at each step with the nearest TPC cluster provided that it fulfils a proximity cut
  - Continue until the inner radius of the TPC is reached
  - Preliminary particle id. based on TPC dE/dx (used for energy loss correction in next steps)
- Track **prolongation to the ITS**
- **ITS standalone tracker:** find tracks from ITS points not attached to TPC prolongations

Tracking: 2nd phase -> outward



- **Back propagation** of tracks with the Kalman filter
  - From the interaction vertex to the TPC outer radius, using the clusters found in previous step
  - Update particle id. based on TPC dE/dx (used for energy loss correction in next step)
- **Prolongation** to TRD, TOF, HMPID, EMCAL, PHOS
  - Association of reconstructed points in these detectors
  - Track length calculation
- *ITS standalone tracks* propagated only up to the outer ITS radius

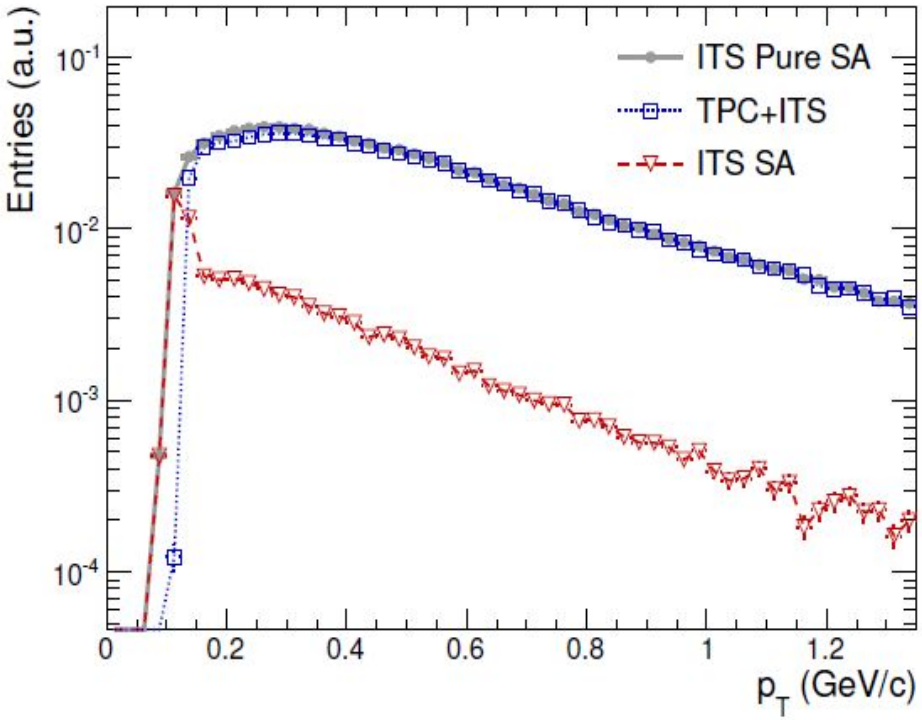
Tracking: 3rd phase -> refit inward



- **Re-fit** of the track in the **inward** direction with the Kalman filter
  - Clusters from the previous steps used in the refit
  - Best determination of the track parameters and covariance matrix
- Tracks propagated to their Distance of Closest Approach (DCA) to the SPD vertex

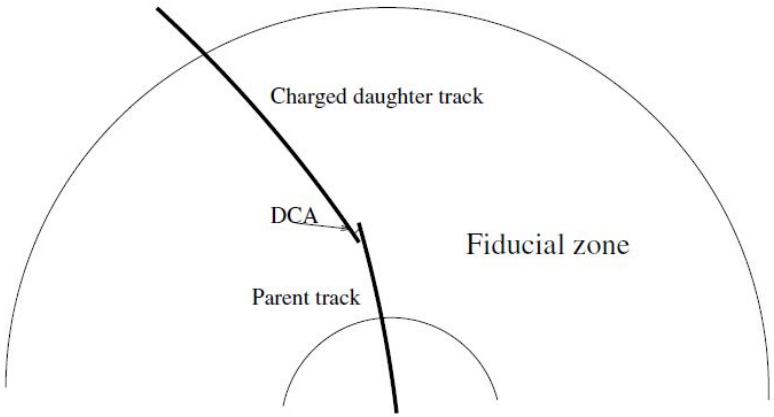
Tracks: global, ITS standalone

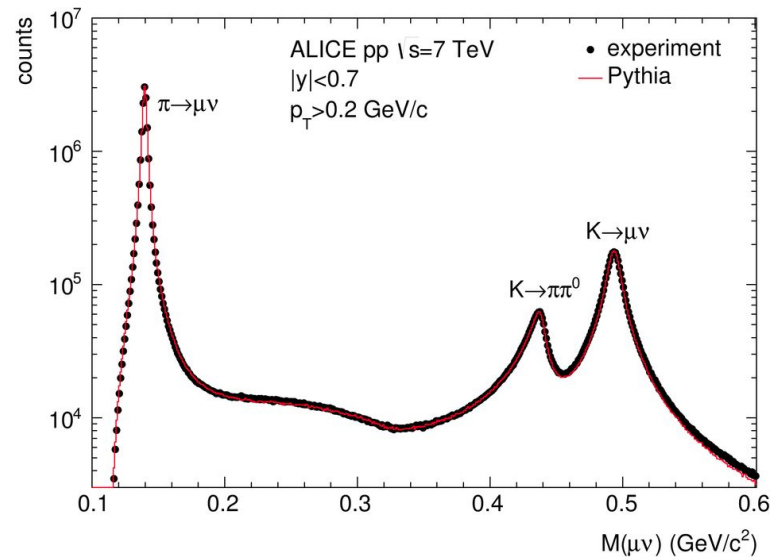
- **Global (TPC+ITS) tracks** = TPC tracks prolonged to ITS with successful refit inward step
- **ITSsa** = ITS only tracks, built from ITS clusters not attached to TPC prolongations
  - Complementary to global tracks
  - Recover: low-p_T tracks not reconstructed in TPC, high-p_T tracks in regions between TPC sectors
- **ITS pure SA**: ITS-only tracks built using all ITS clusters
  - Enabled only in pp and p-Pb
  - For checks+specific analyses



- **NOTE:** ITS-only tracks have worse p_T resolution than TPC+ITS

Kinks

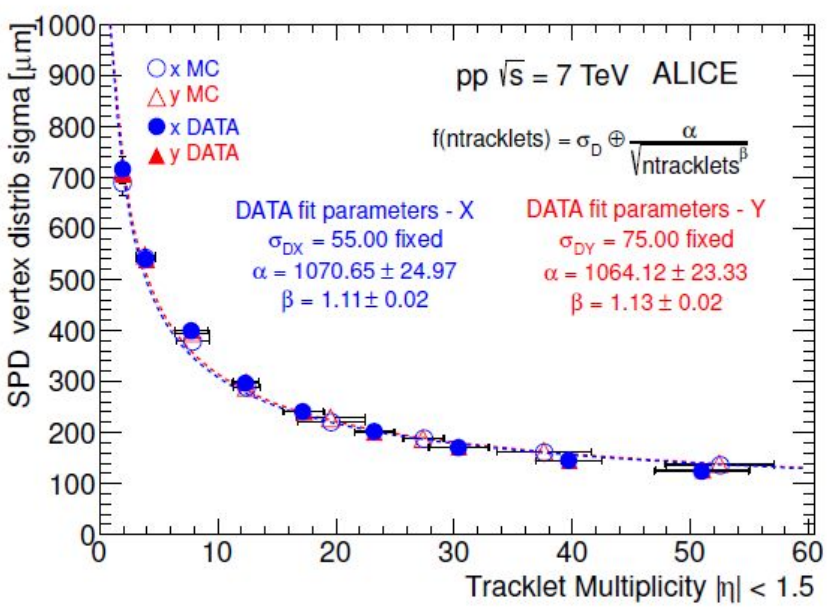


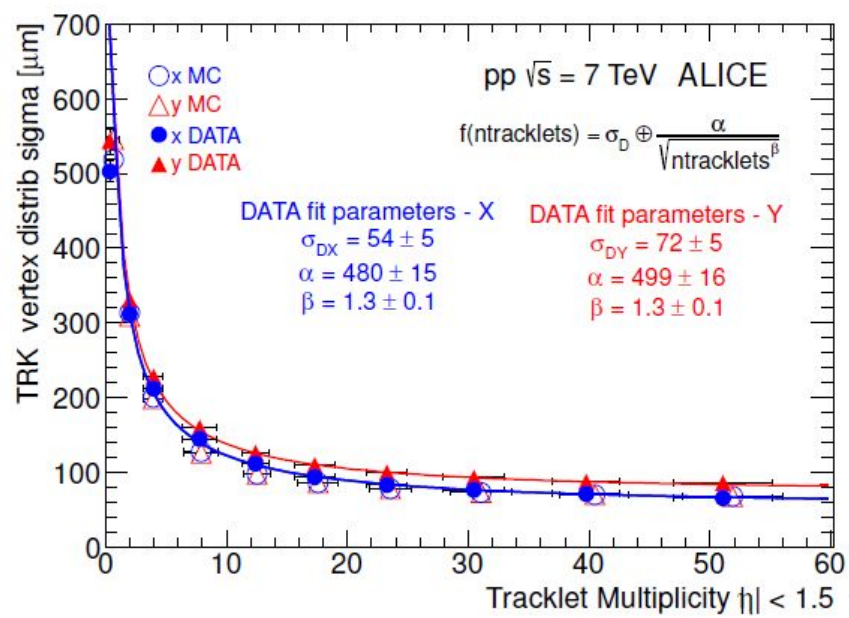


- **Kink** = topological signature of charged particles decaying into 1 charged + 1 neutral particles (e.g. K→μν, π→μν)
- Search for kinks inside the volume of the TPC after the first inward tracking step
  - Large decay angle kinks = pairs of tracks with same charge that intersect each other in a fiducial volume
  - Small decay angle kinks = "breakpoint" (i.e. change in direction) between the upstream and downstream parts of a track

Primary vertex from tracks

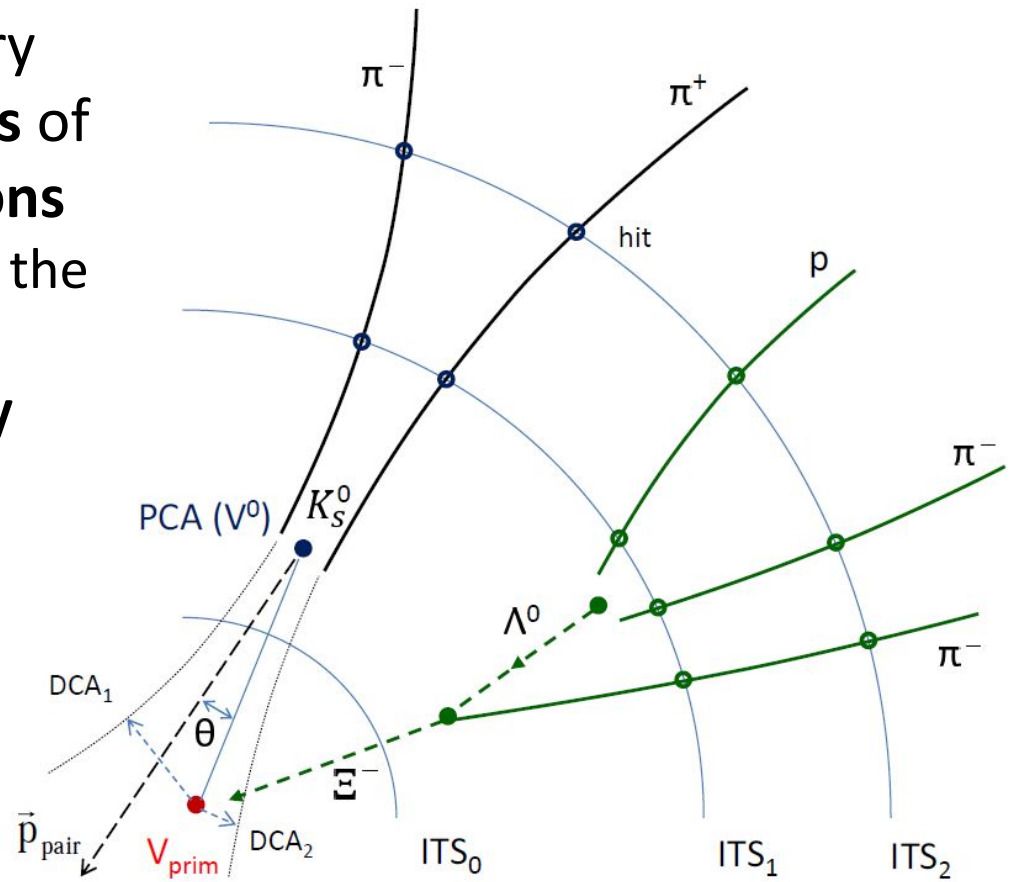
- **Primary vertex** determined from reconstructed **TPC+ITS tracks**
  - Better resolution than SPD vertex
  - Lower efficiency (at low multiplicity) than SPD vertex
  - Pileup tagging via multiple vertices (*MultiVertex*)
- Primary vertex also determined from track parameters using only TPC information (**TPC-only**)
  - Poor resolution, not to be used in analysis





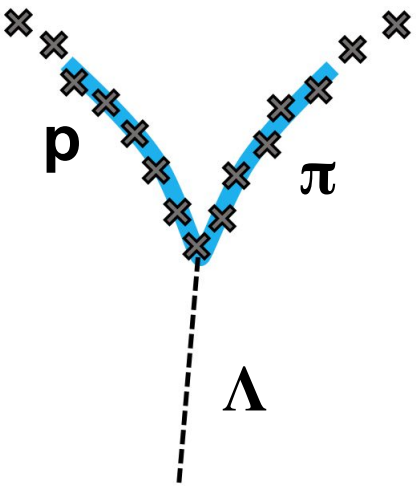
Secondary vertices

- Search for **photon conversions** and secondary vertices from **weak decays** of long-lived (**strange**) **hadrons**
  - Tracks with **large DCA** to the primary vertex
  - Select on **decay topology**



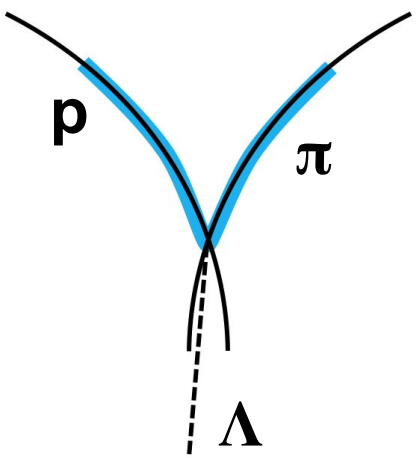
- V0 candidates:
  - $K^0_s \rightarrow \pi\pi$
  - $\Lambda \rightarrow p\pi$
  - $\gamma \rightarrow ee$
- Cascade candidates
  - $\Xi \rightarrow \Lambda\pi \rightarrow p\pi\pi$

V0 candidates: Λ and K^0_S



On-the-fly V0s: clusters -> V0 candidate

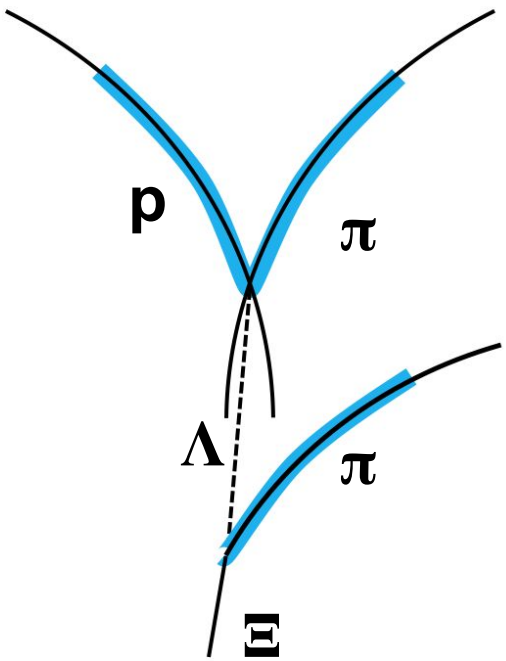
- V0s found during tracking
- **Good**: Raw cluster info used: ideal resolution + causality cut (no clusters before decay point)
- **Bad**: cannot be replayed at analysis level e.g. with looser selections



Offline V0s: Tracks -> V0 candidate

- V0s found based on reco'ed tracks
- **Good**: high level of abstraction: runs on ESD tracks and can be replayed and reconfigured at will
- **Bad**: less-than-ideal resolution because of cluster association before decay point

Cascade candidates: Ξ and Ω

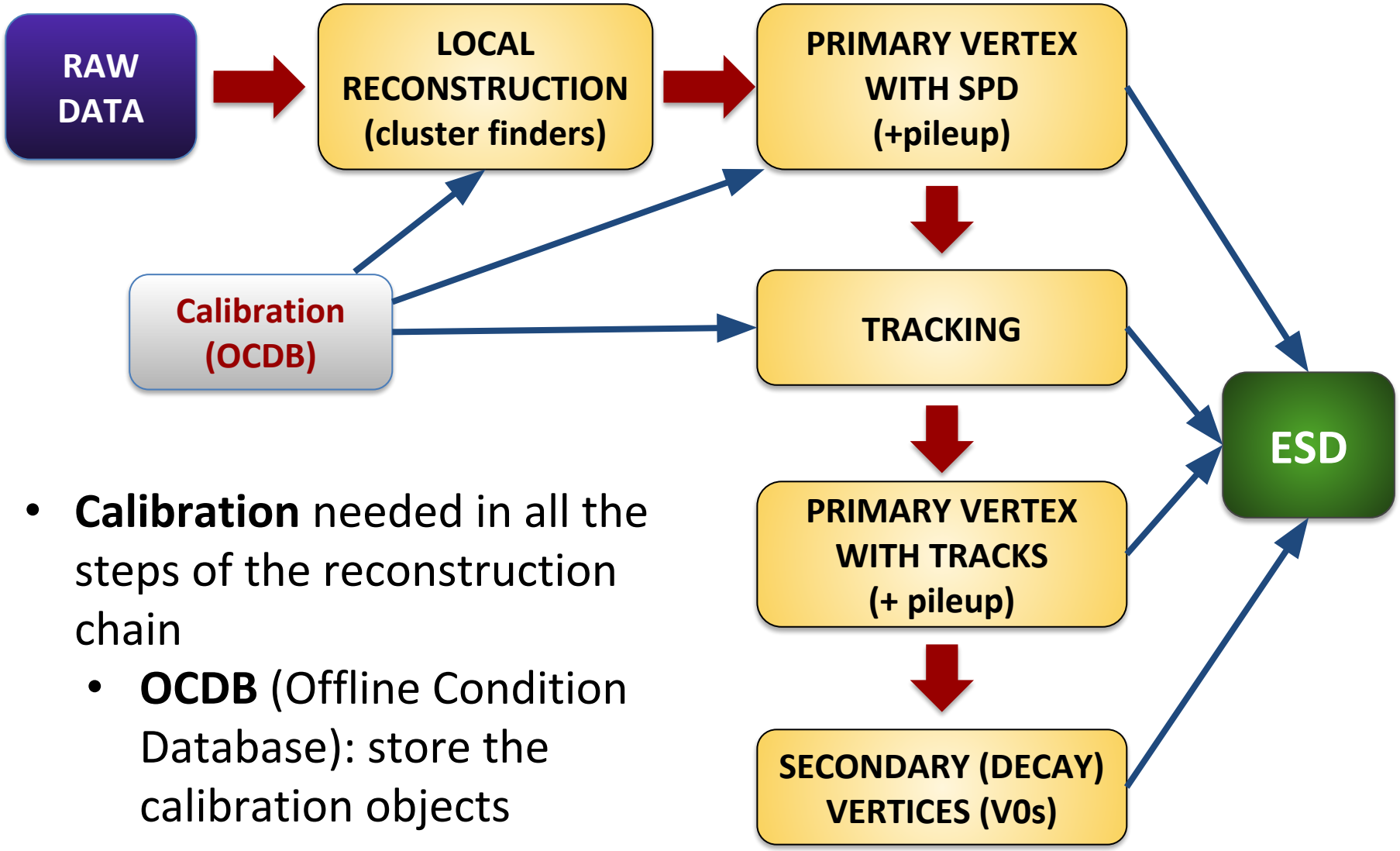


Cascades: V0 + track -> cascade candidate

- Only cascades based on offline V0s have been used in ALICE so far
- **Same strategy as V0s**, but larger number of topological selections (total: 10 selections)
- **A selection on Λ invariant mass** is also performed (typically we select a window of 6-10 MeV/c^2 around the PDG mass)
- However, for the cascade invariant mass calculation, **we use the perfect Λ mass** for the V0 daughter

RAW DATA PRODUCTION CHAIN

General reconstruction strategy



- **Calibration** needed in all the steps of the reconstruction chain
  - **OCDB** (Offline Condition Database): store the calibration objects

Calibrations

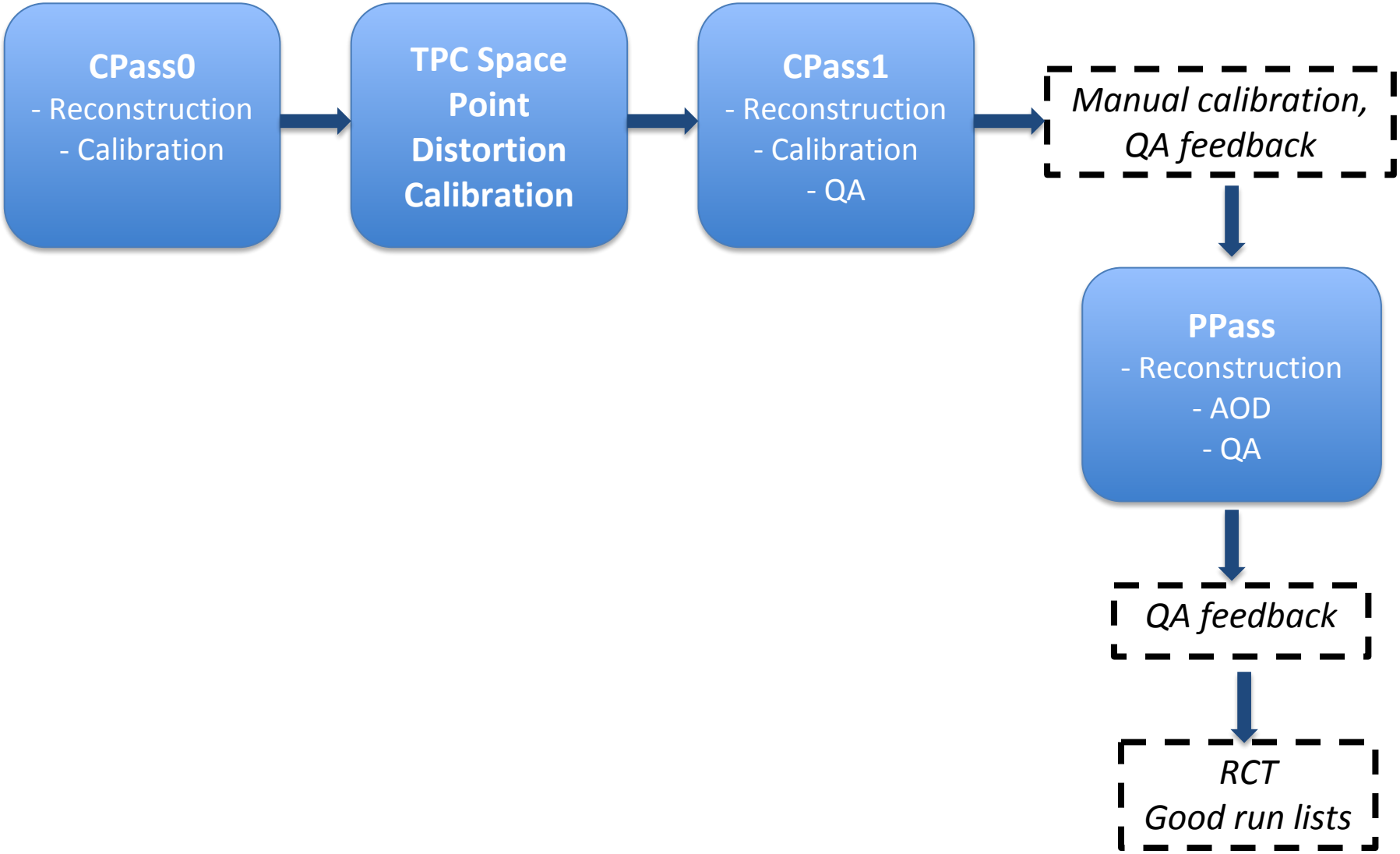
- The reconstructed data quality (resolution, minimisation of biases, particle identification …) critically depends on the quality of the **calibrations** used in the reconstruction, e.g.:
  - Maps of *dead and noisy elements*, *gain*, signal amplitudes
  - For drift detectors (SDD, TPC): *drift velocity*, drift field maps, *distortions*...
  - Actual position of the detectors (*alignment*)
  - Geometry of the *luminous region*
- **Sources of calibration**:
  - *Online*, via **Shuttle**
    - From dedicated calibration runs, or calibration triggers or also from interaction events
  - *Offline*, via **automatic** calibration passes (**CPass0, CPass1**)
  - *Offline*, via "**manual analysis**"

Online calibration

- At the end of the run (physics or special calibration run) a **Detector Algorithm** (DA) is run, the online calibration parameters are produced, collected by the **Shuttle** together with the **DCS data points** and stored in the **OCDB**
- Examples:

System	Condition data	Special runs	Physics runs online
SPD	trigger chip map and thresholds		half-stave status pixel noise
SDD		anode ped (peds) anode gain, status (puls) anode v_{drift} (inject)	
SSD		strip ped, noise, status (peds)	
TPC	$P, T(x, y, z)$ pad status trigger t_0	pad gain (Kr) pad noise (peds) v_{drift} (laser) pad status (puls)	v_{drift} (laser)

Raw data processing steps

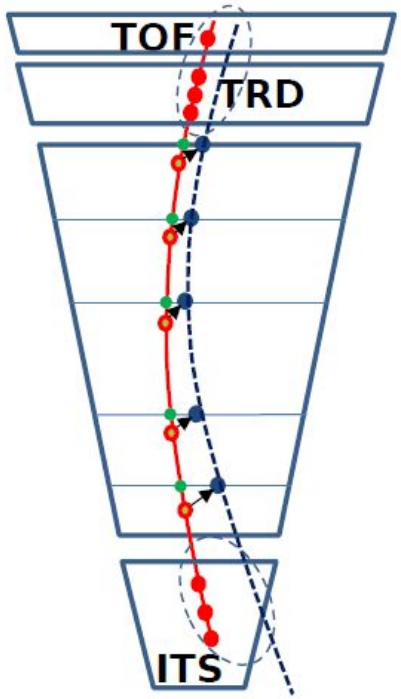


Calibration chain: CPass0

- CPass0 **reconstruction**:
  - Performed run-wise, starts promptly after the data taking
  - Uses events from the *kCalibBarrelMB* trigger alias
    - Typically a mixture of CINT7 and HM triggers
- CPass0 reconstruction is followed by
  - CPass0 **calibration train** (= standard analysis train which runs over the ESDs + friends) and fills calibration objects which are stored in the **OCDB**
    - **TPC**: vdrift, t0, gain
    - **TRD**: vdrift, t0, gain, chamber status, ExB
    - **TOF**: t0 global offset
    - **T0**: t0
    - **Luminous region**: <x>, <y>, <z>, σ_x, σ_y, σ_z
  - **TPC space point distortion** calibrations

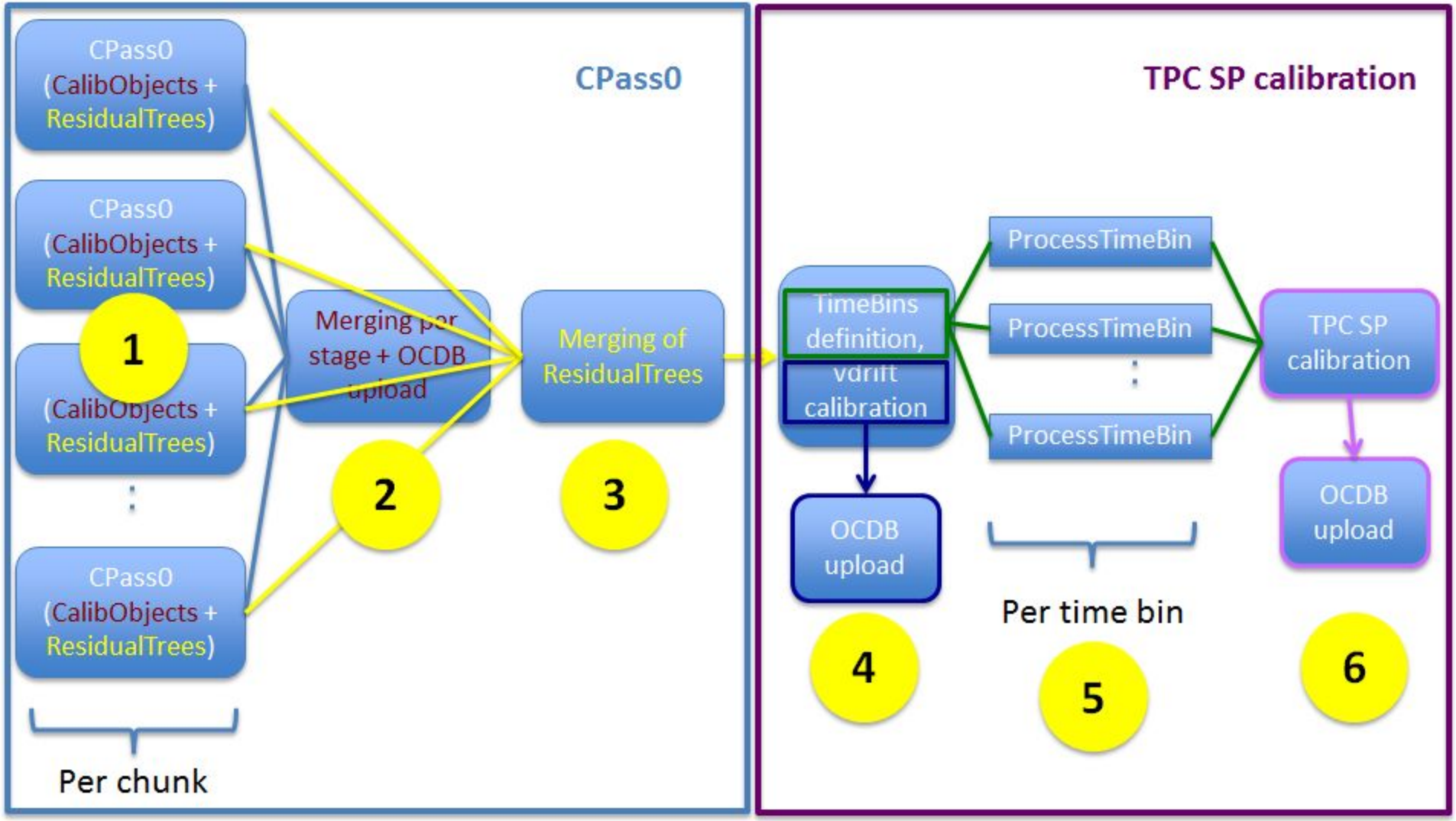
TPC Space Point distortions

- Distortions in the TPC drift field due to space charge
  - More pronounced in Run2 with Ar gas (2015-2016)
  - Up to O(5 cm) effect on the reconstructed space points
  - Need to be corrected, target precision O(0.2 mm)
- Procedure:



1. TPC reconstruction with large road-widths to not loose TPC clusters attachment
2. Match to ITS and TRD/TOF
3. Refit ITS-TRD-TOF part and interpolate to TPC as a **reference** of **true track** at every pad-row
4. Collect Y, Z differences between **distorted clusters** and **reference** points in sub-volumes (voxels) of TPC
5. Extract 3D vector of distortion in every voxel
6. Create smooth parameterization (DB object) to use for correction during following reconstruction
7. Distortions change with time (interaction rate): procedure in short time intervals (~20-40 min, restriction by statistics)

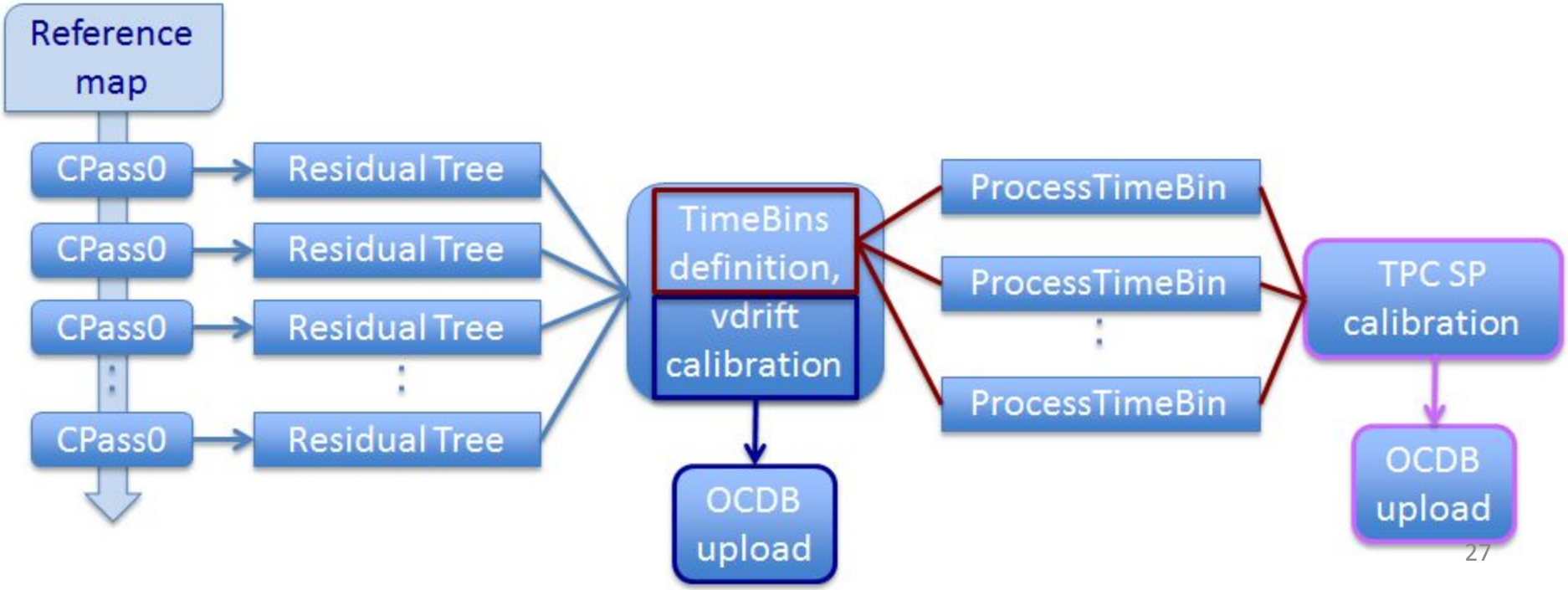
CPass0 -> TPC SP calibration



NOTE: numbers in yellow boxes represent the jobs that run on the grid

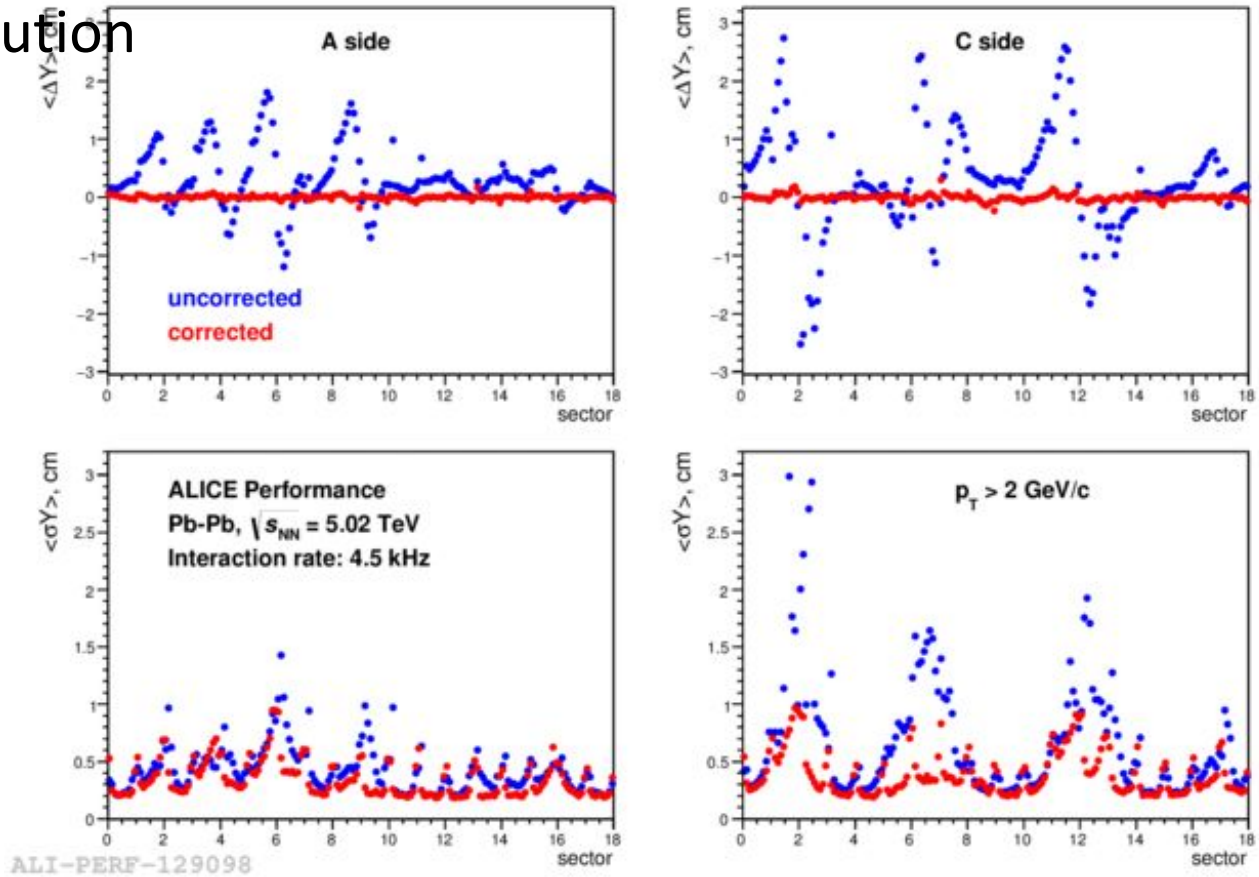
TPC SP distortions vs. luminosity

- Distortions scale ~linearly with luminosity
- Granularity of the time-bin is O(20-40mins)
  - Enough to guarantee ~constant distortions within one time-bin
- **Reference map** scaled with luminosity are used in CPass0 to reduce the distortions we start with
  - Reference map created from special runs at different IR and B field

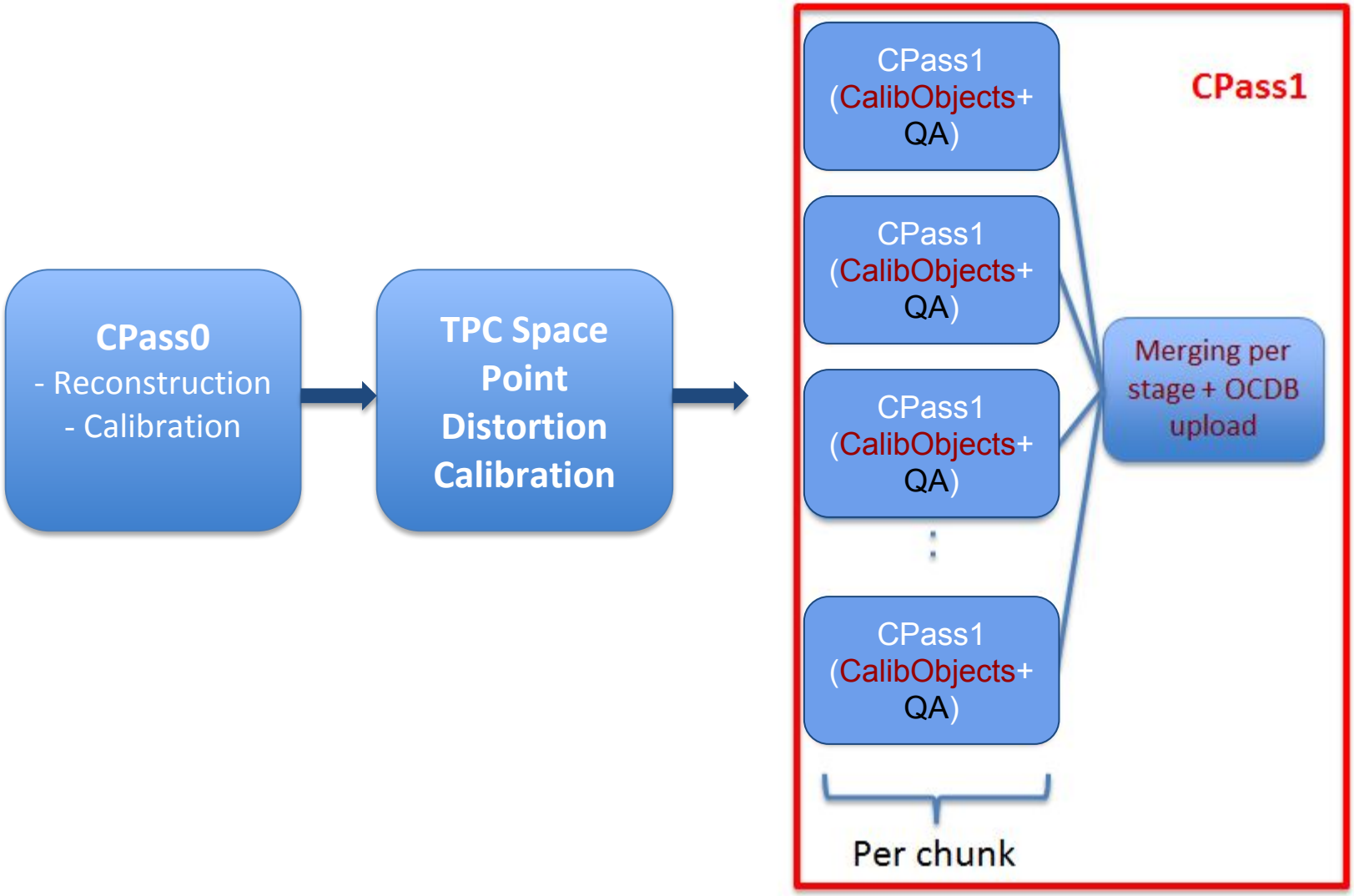


TPC Space Point distortions

- Distortions in the TPC drift field due to space charge
- After correction: bias on the Distance of Closest Approach of the TPC track to the vertex reduced below intrinsic resolution



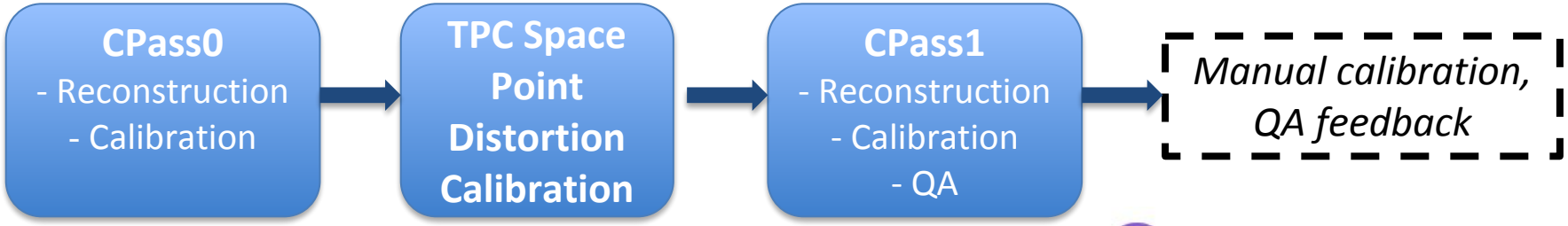
Calibration chain: CPass1



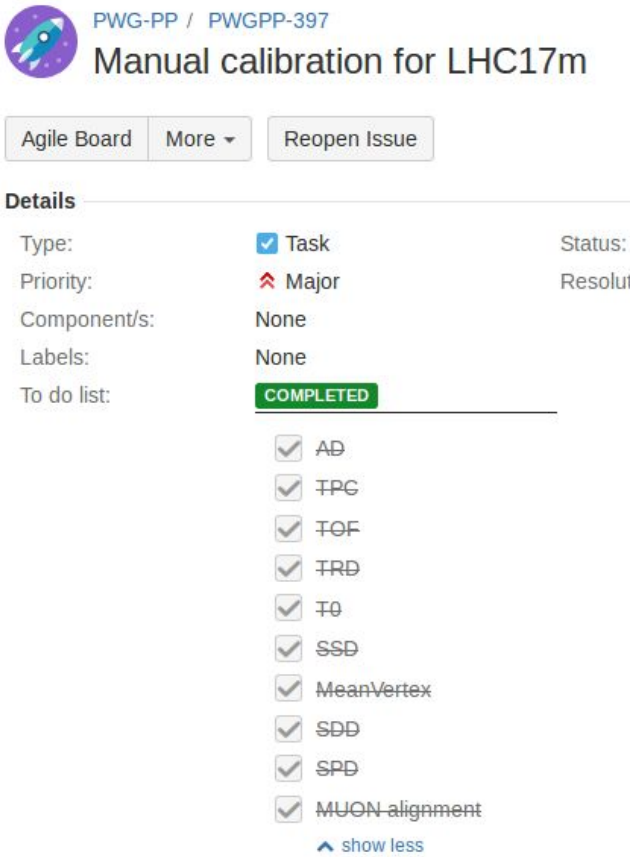
Calibration chain: CPass1

- CPass1 **reconstruction**:
  - Performed run-wise, starts after the completion of CPass0
  - Uses events from the *kCalibBarrel* trigger alias
    - Typically a subsample (~10%) of CINT7 triggers
- CPass1 **goals:**
  - *Refined calibrations* after reconstruction with TPC calibrated from CPass0 and TPC SP distortion corrections
  - *Quality assurance* of the detector calibrations extracted from CPass0
    - The calibrations from CPass1 and manual calibrations are checked in the QA of the subsequent PPass
- CPass1 reconstruction followed by:
  - **Calibration train**: fill calibration objects + store in **OCDB**
  - **QA train** -> produce root files with QA information

After CPass1: QA+manual calibration

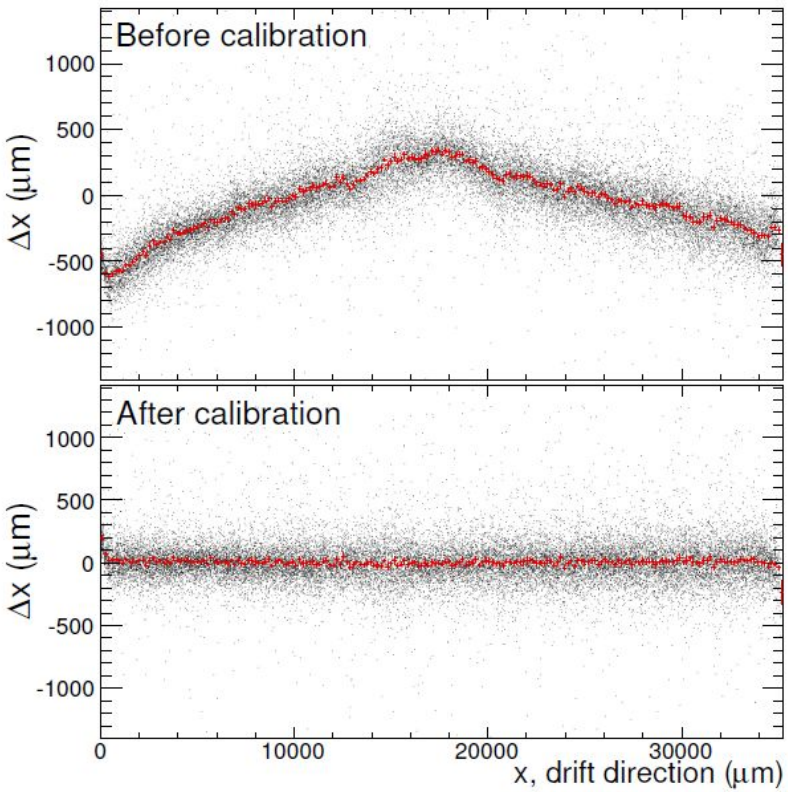


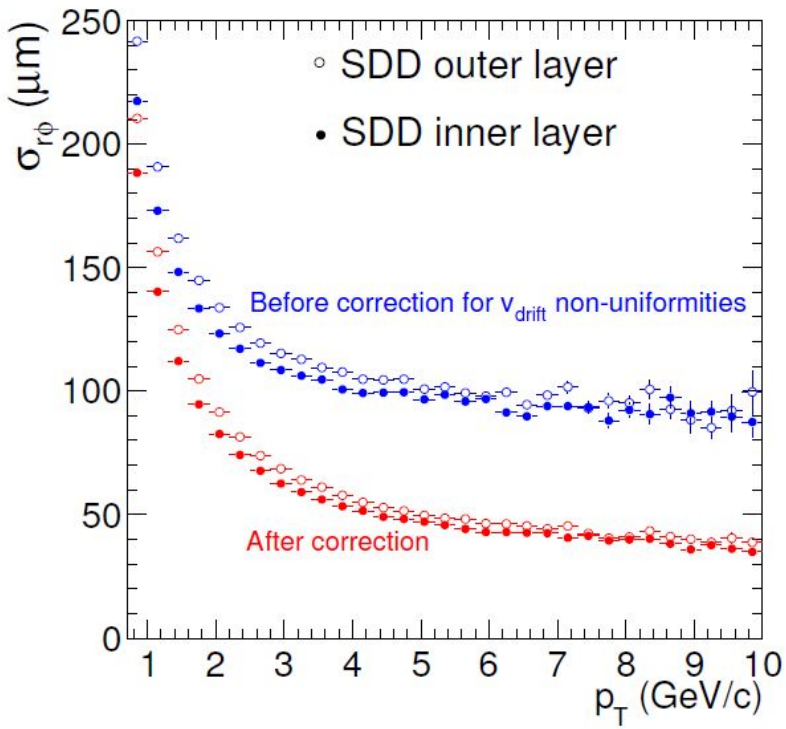
- After CPass0/1 is done for **all runs of a period**, a ~4-day window is opened for
  - **Quality Assurance** checks
  - **Manual calibration** for:
    - Calibrations that need more statistics than that of a single run
      - E.g. TOF channel-by-channel t0 offset when needed
    - Smoothen/fix calibrations for runs affected by fluctuations
      - E.g. luminous region



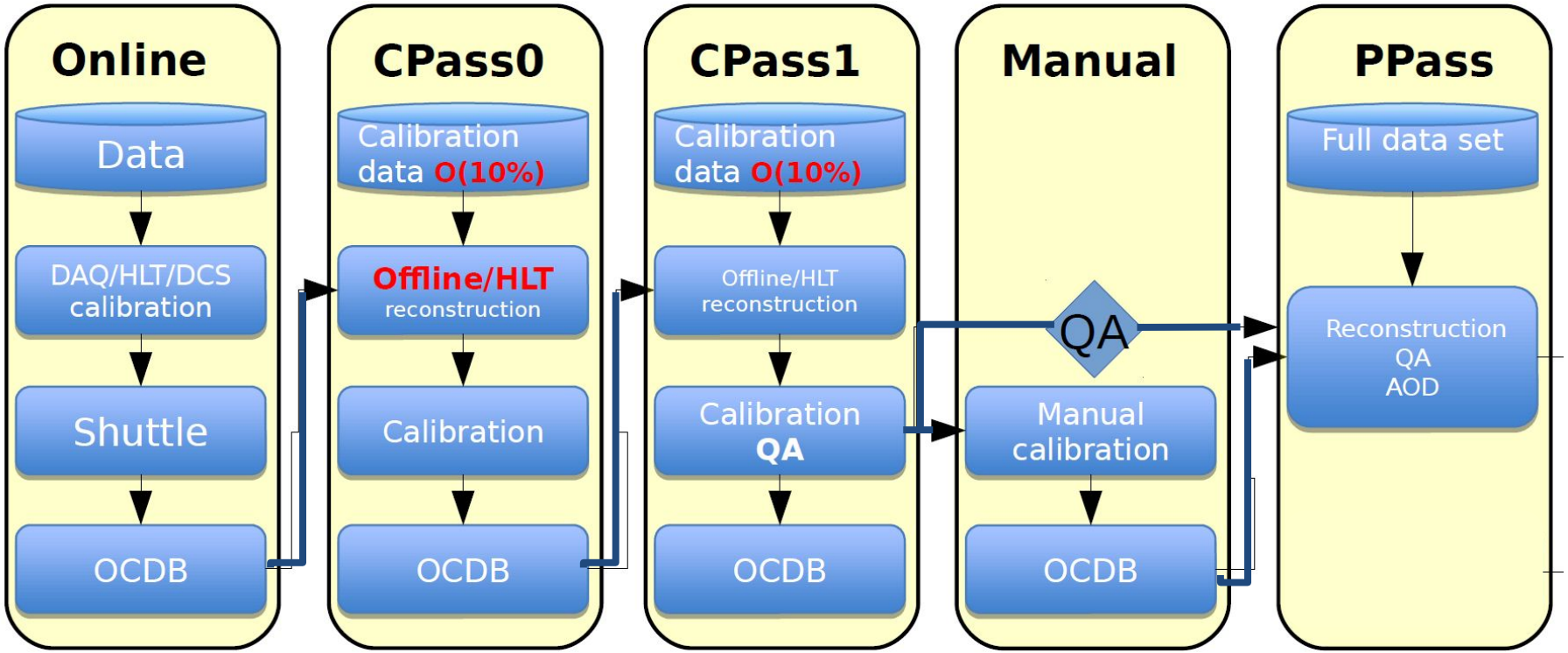
Example: SDD manual calibration

- Calibration at fill level of:
  - Drift velocity residual calibration + time0 + map for non-uniformities of the drift field
  - ADC -> keV calibration for dE/dx in bins of drift time



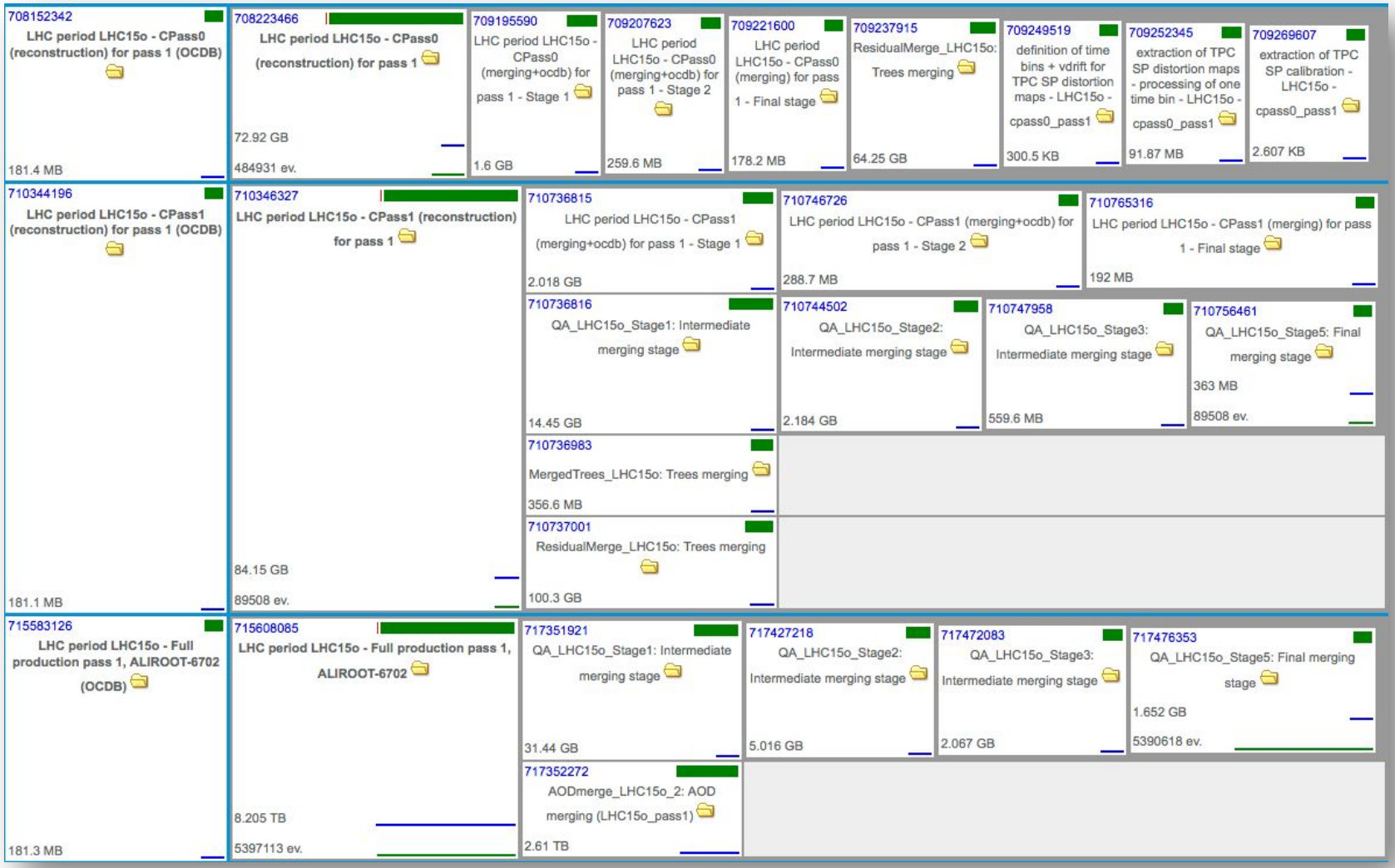


Reconstruction chain: summary



DPG ALICE • Data Preparation Group

Raw data processing chain: jobs



Each box is a grid job → several (27) masterjobs per run in the chain

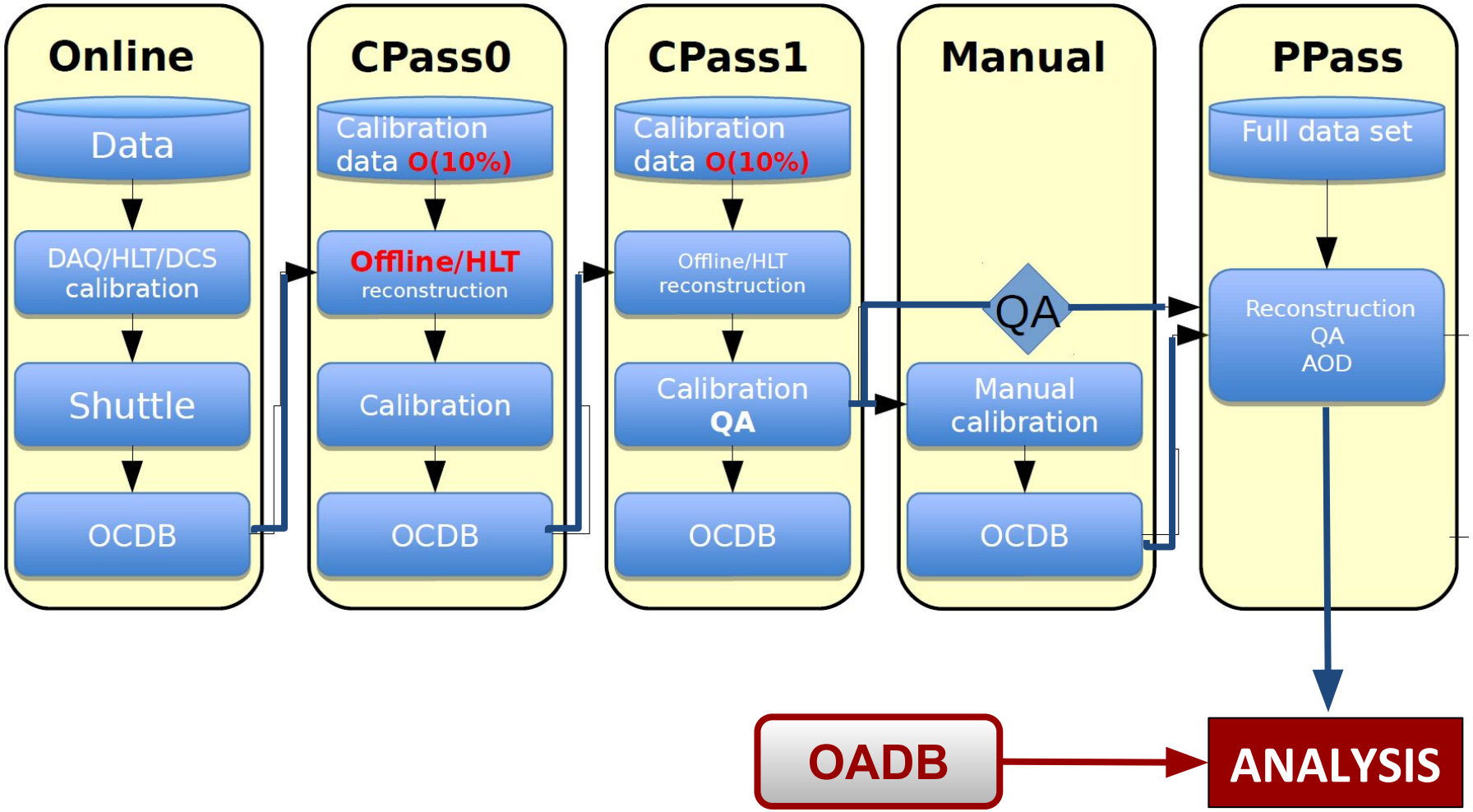
muon_calo reconstruction

- Aim: fast reconstruction for detectors that do not need the automatic calibration of CPass0/1
- Involved detectors:
  - MUON, EMCAL, PHOS, T0, ITS, V0, ZDC, AD

```
// Set reconstruction flags (skip detectors here if neded with -<detector name>
rec.SetRunReconstruction("ALL -TPC -TRD -TOF -HLT -PMD");
```

- Performed for all good PHYSICS runs in the logbook
- Starts promptly after the data taking
  - In parallel with CPass0/1
- QA performed in parallel with CPass1 QA

Offline Analysis Data Base

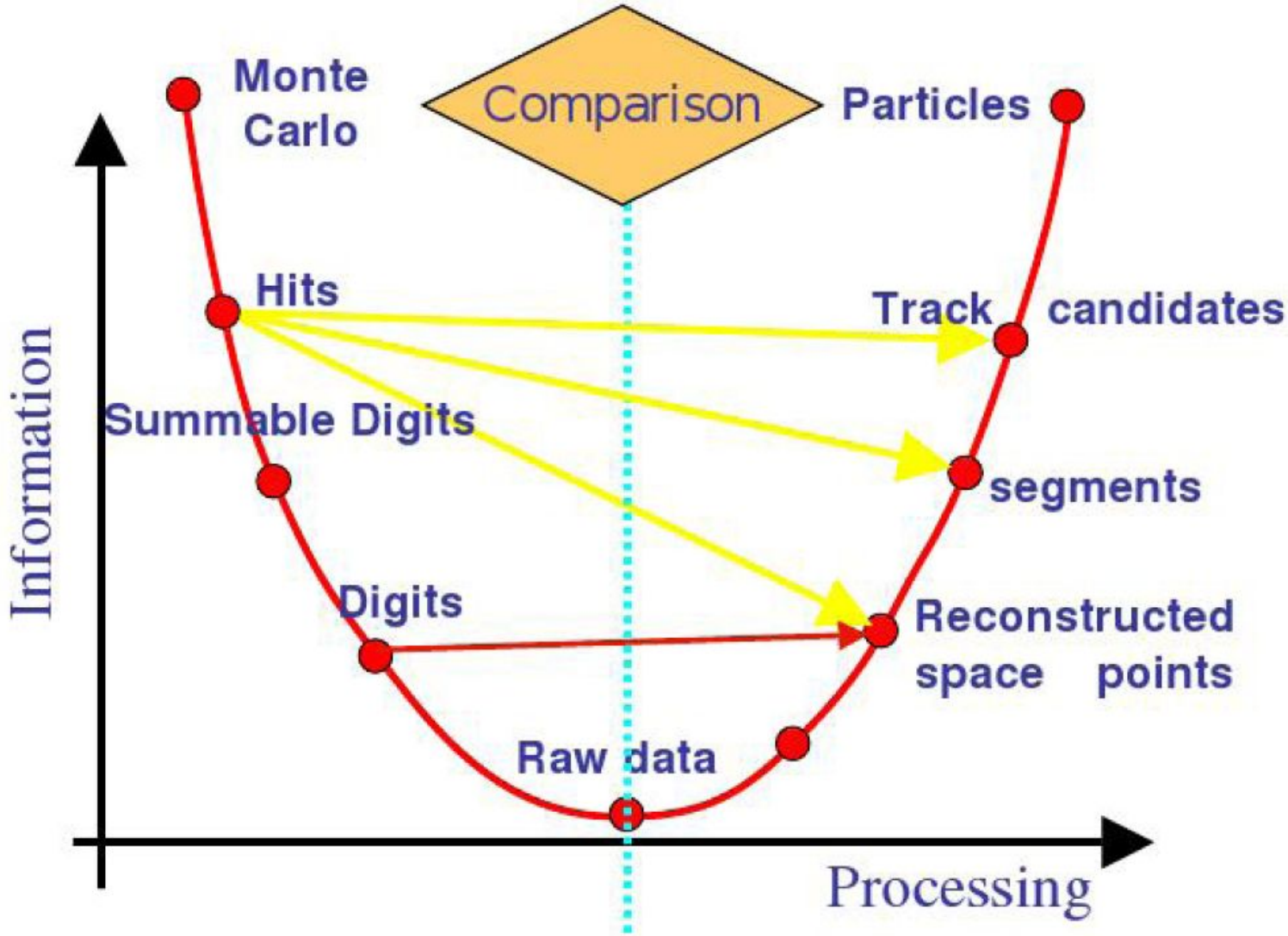


Offline Analysis Data Base

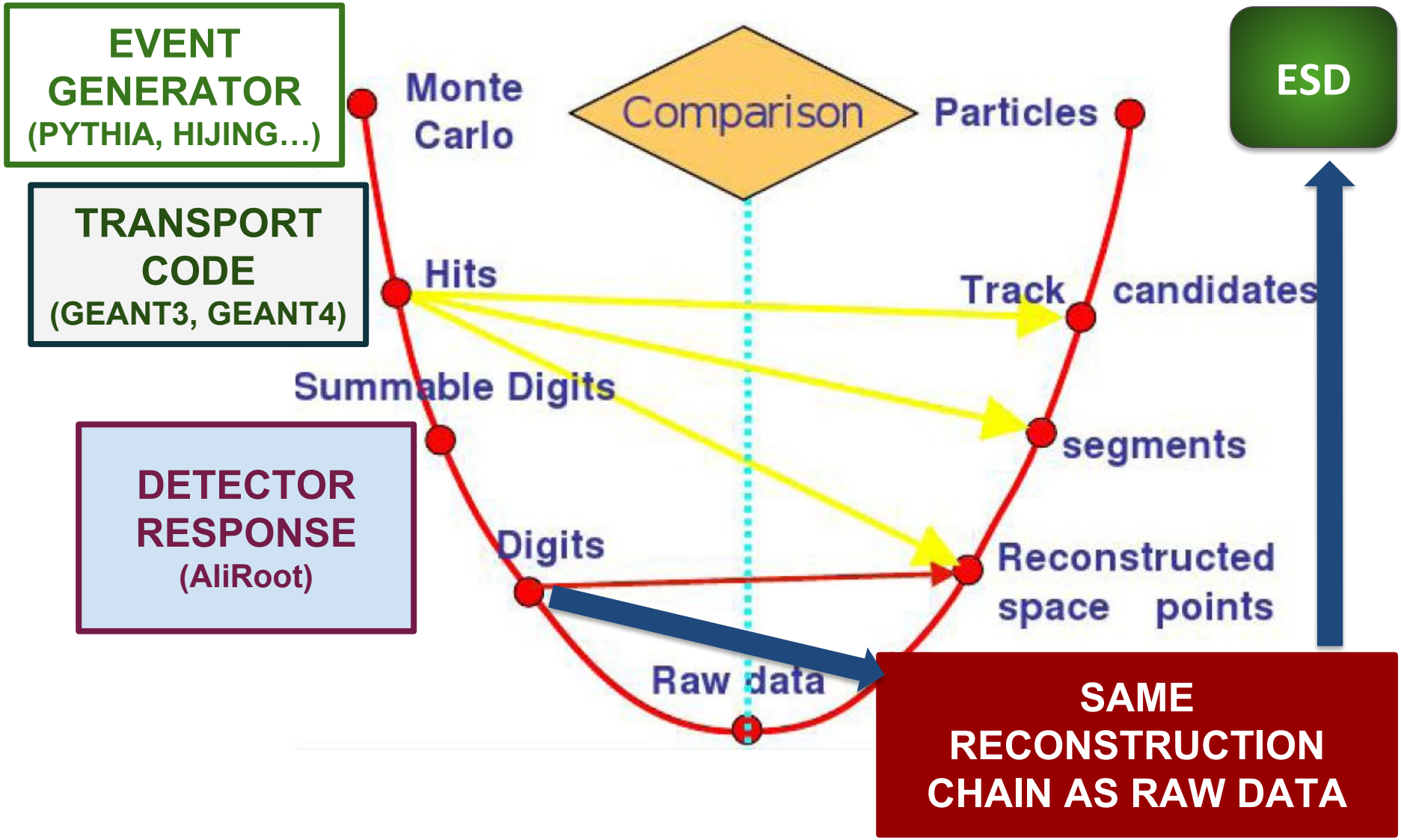
- Calibrations needed at the analysis level, e.g.
  - **Splines** to parametrise the expected **TPC dE/dx** for particle species
    - https://twiki.cern.ch/twiki/bin/view/ALICE/TPCSplines
  - **Centrality/multiplicity** percentiles from estimators
    - https://twiki.cern.ch/twiki/bin/viewauth/ALICE/AliMultSelectionCalibStatus
  - **Event plane** flattening corrections
- Calibration files produced asynchronously w.r.t. reconstruction pass
  - Usually slightly after muon_calo or PPass completion
- Stored in **OADB**
  - Directories in AliPhysics, under $ALICE_PHYSICS/OADB
  - Need to use recent enough AliPhysics tag in your analysis if you need latest calibrations

MONTE CARLO SIMULATIONS

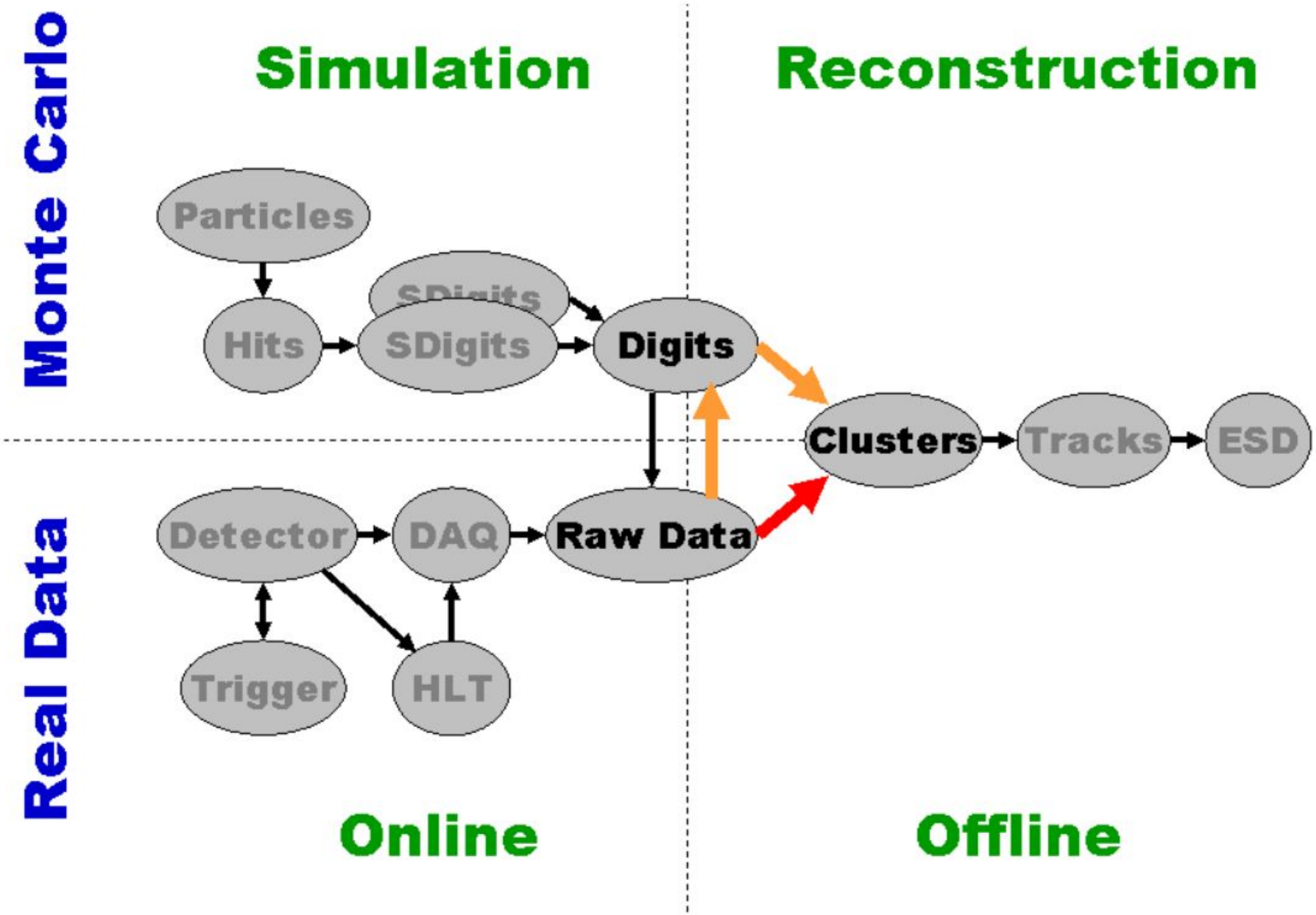
Monte Carlo chain



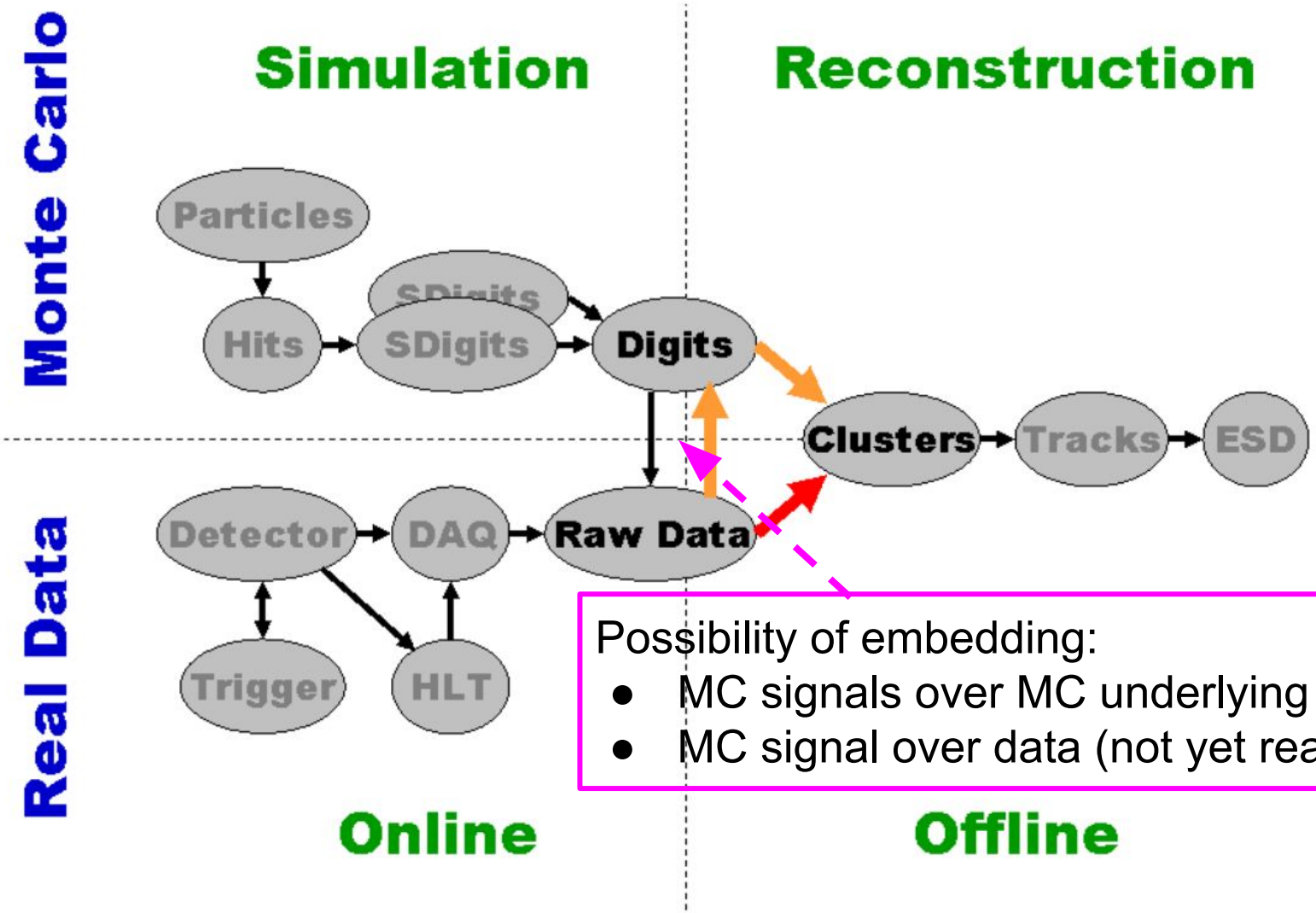
Monte Carlo chain



Monte Carlo vs. raw data chain



Monte Carlo vs. raw data chain

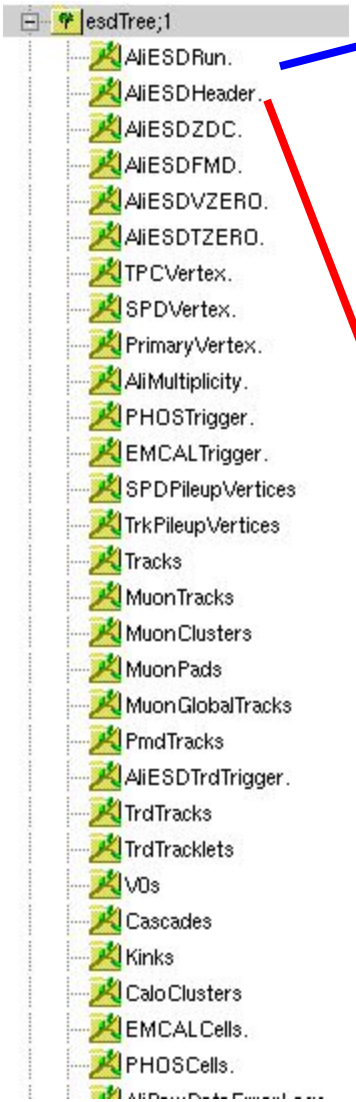


Possibility of embedding:

- MC signals over MC underlying event
- MC signal over data (not yet ready)

ESD EVENTS AND TRACKS

AliESDs.root contents (1)



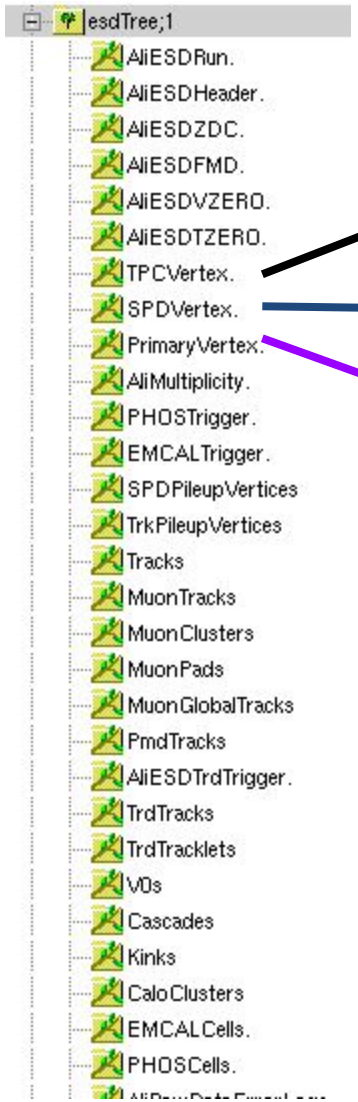
ESD run

```
Float_t        fCurrentL3;       // signed current in the L3     (LHC co
Float_t        fCurrentDip;      // signed current in the Dipole (LHC co
Float_t        fBeamEnergy;      // beamEnergy entry from GRP
Double32_t     fMagneticField;   // Solenoid Magnetic Field in kG : for
Double32_t     fMeanBeamInt[2][2]; // mean intensity of interacting and
Double32_t     fDiamondXY[2];    // Interaction diamond (x,y) in RUN
Double32_t     fDiamondCovXY[3]; // Interaction diamond covariance (x,y)
Double32_t     fDiamondZ;        // Interaction diamond (z) in RUN
Double32_t     fDiamondSig2Z;    // Interaction diamond sigma^2 (z) in R
UInt_t         fPeriodNumber;    // PeriodNumber
Int_t          fRunNumber;       // Run Number
Int_t          fRecoVersion;     // Version of reconstruction
Int_t          fBeamParticle[2]; // A*1000+Z for each beam particle
TString        fBeamType;        // beam type from GRP
```

ESD header

```
// Event Identification
ULong64_t    fTriggerMask;       // Trigger Type (mask) 1-50 bits
ULong64_t    fTriggerMaskNext50; // Trigger Type (mask) 51-100 bits
UInt_t       fOrbitNumber;       // Orbit Number
UInt_t       fTimeStamp;         // Time stamp
UInt_t       fEventType;         // Type of Event
UInt_t       fEventSpecie;       // Reconstruction event specie (1-
UInt_t       fPeriodNumber;      // Period Number
Int_t        fEventNumberInFile; // Running Event count in the file
UShort_t     fBunchCrossNumber;  // Bunch Crossing Number
UChar_t      fTriggerCluster;    // Trigger cluster (mask)
```

AliESDs.root contents (2)



PRIMARY VERTEX FROM TPC

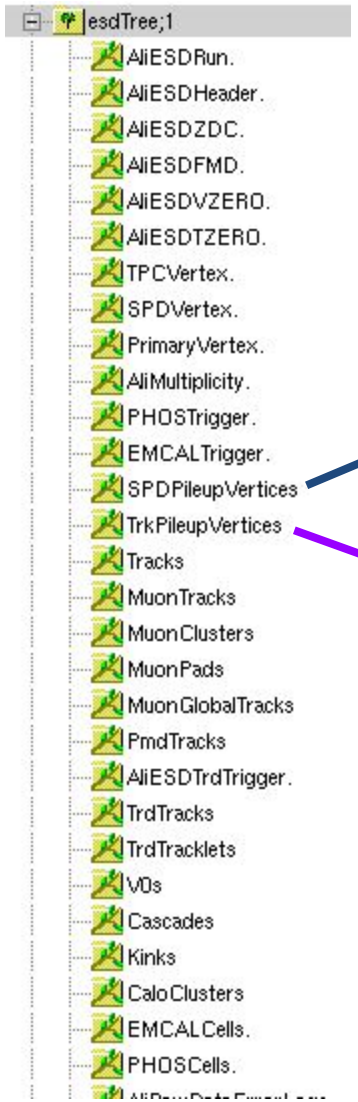
PRIMARY VERTEX FROM SPD

PRIMARY VERTEX FROM TPC+ITS TRACKS

NOTE on AliESDEvent::GetPrimaryVertex

```
const AliESDVertex * AliESDEvent::GetPrimaryVertex() const
{
  //
  // Get the "best" available reconstructed primary vertex.
  //
  if(fPrimaryVertex){
    if (fPrimaryVertex->GetStatus()) return fPrimaryVertex;
  }
  if(fSPDVertex){
    if (fSPDVertex->GetStatus()) return fSPDVertex;
  }
  if(fTPCVertex) return fTPCVertex;
```

AliESDs.root contents (3)



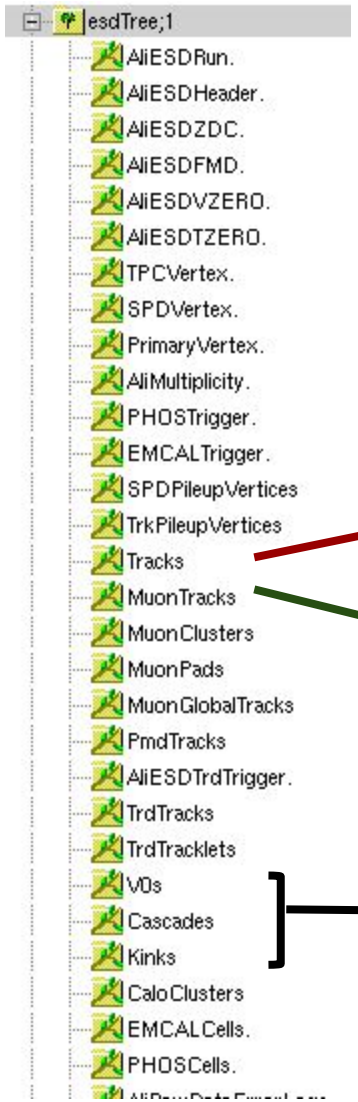
ARRAY OF PILEUP VERTICES FROM SPD

ARRAY OF PILEUP VERTICES FROM TPC+ITS TRACKS (MultiVertex)

+ PILEUP-TAGGING METHODS

```
virtual Bool_t  IsPileupFromSPD(Int_t minContributors=5,
                                Double_t minZdist=0.8,
                                Double_t nSigmaZdist=3.,
                                Double_t nSigmaDiamXY=2.,
                                Double_t nSigmaDiamZ=5.) const;

virtual Bool_t IsPileupFromSPDInMultBins() const;
```

AliESDs.root contents (4)



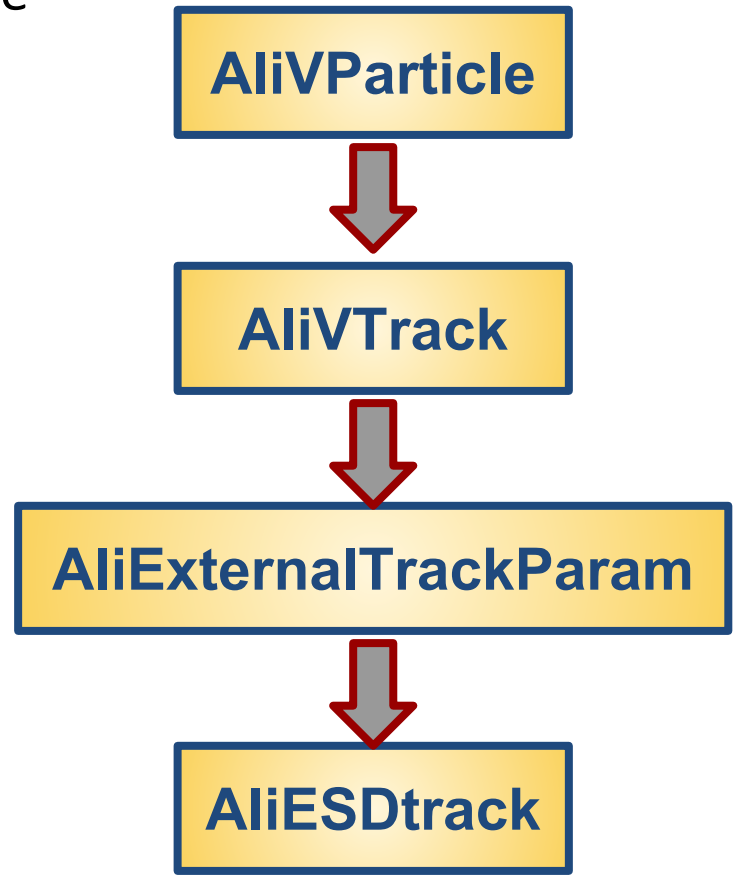
ARRAY OF AliESDtracks (CENTRAL BARREL)

ARRAY OF AliESDMuonTracks (MUON ARM)

ARRAYS OF V0s, CASCADES, KINKS

AliESDtrack

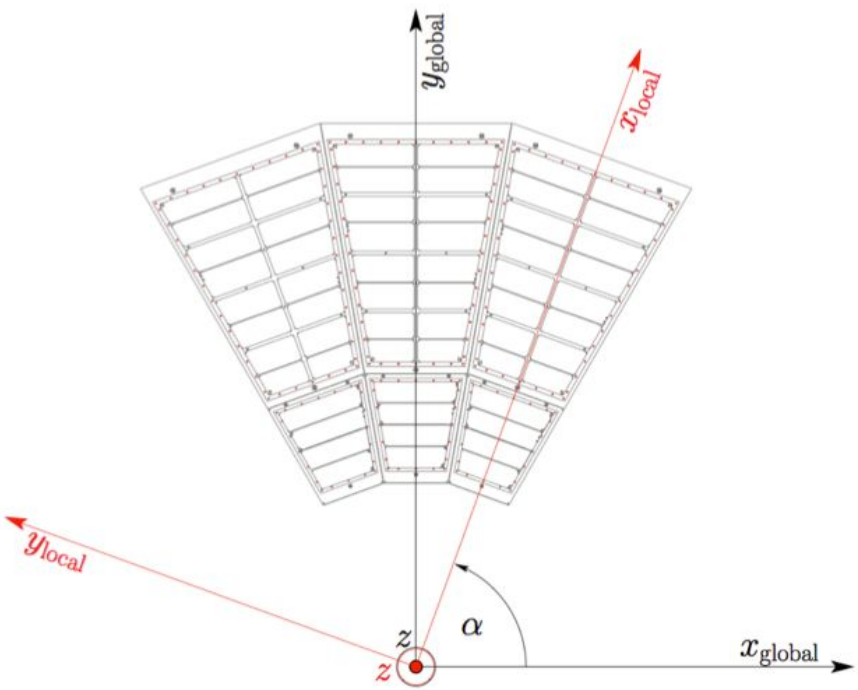
- Inherits from **AliExternalTrackParam**:
  - Store track parameters and covariance matrix
  - Methods to get momentum, position impact parameter …
  - Methods to "propagate" the track along its trajectory
- Additional information on track properties, e.g.:
  - Status bit (TPC in, …., ITS refit)
  - chi2 of track fit (per detector)
  - Number of attached clusters (per detector), TPC findable clusters, ITS per-layer info, ...
  - PID related information (track length, TOF time, EMCAL cluster…)

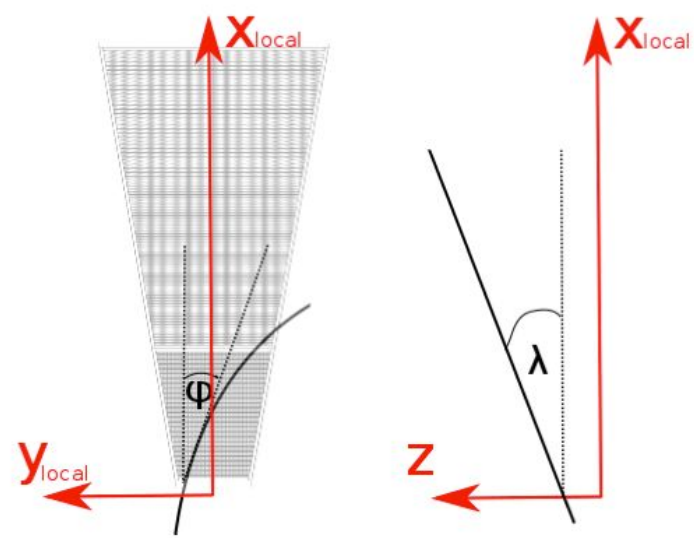


Track parameterisation

- STEER/STEERBase/AliExternalTrackParam
- Track parameters in the **“local” reference system**

```
Double32_t          fX;     // X coordinate for the point of parametrisation
Double32_t          fAlpha; // Local <-->global coor.system rotation angle
Double32_t          fP[5];  // The track parameters
Double32_t          fC[15]; // The track parameter covariance matrix
```



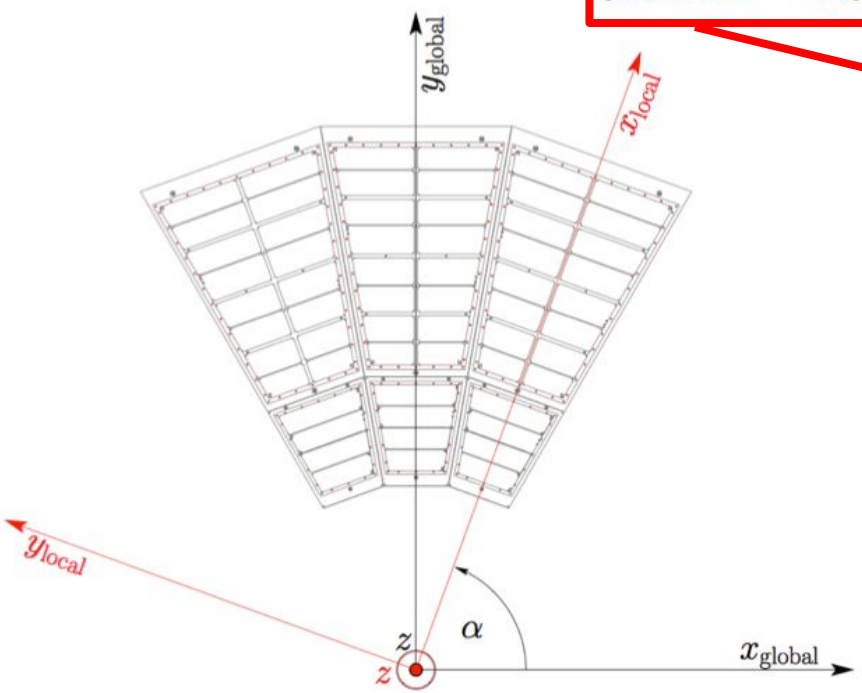


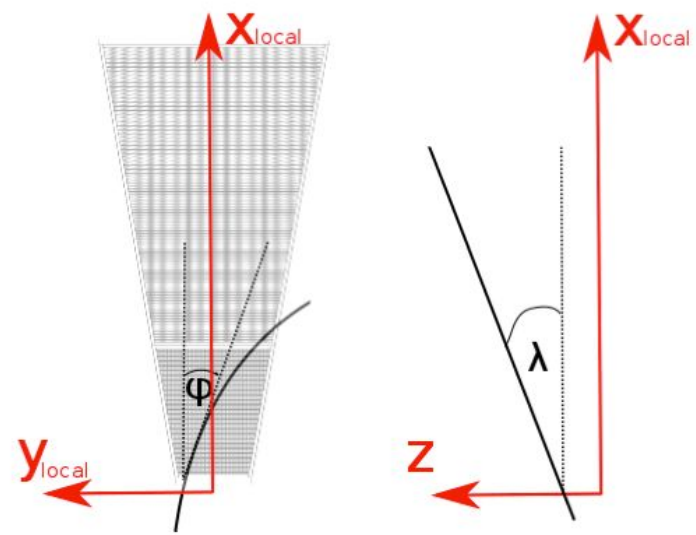
Track parameterisation

```
Double32_t          fX;     // X coordinate for the point of parametrisation
Double32_t          fAlpha; // Local <-->global coor.system rotation angle
Double32_t          fP[5],  // The track parameters
Double32_t          fC[15]; // The track parameter covariance matrix
```

```
param0:   local Y-coordinate of a track (cm)
param1:   local Z-coordinate of a track (cm)
param2:   local sine of the track momentum azimuthal angle
param3:   tangent of the track momentum dip angle
param4:   1/pt (1/(GeV/c))
```

$$(y, z, \sin(\varphi), \tan(\lambda), 1/p_t)$$





Parameterizations in AliESDtrack

- **Main parameterization** is **at the DCA to the primary vertex** (note: primary vertex not included in the track fit)
  - **Optimal** for **primary particles**
- Five additional parameterizations (snapshots along the trajectory at different tracking steps) stored in AliESDtrack:
  - ***GetConstrainedParam()*** = track parameters constrained at the primary vertex **-> used for analyses that need uniform φ coverage + optimal resolution for primaries (e.g. jets)**
  - ***GetInnerParam()*** = Track parameters at the inner wall of the TPC (from refit step) **-> used for TPC pid**
  - ***GetTPCInnerParam()*** = TPC-only parameters propagated at DCA to the primary vertex, corrected for material between TPC and vertex (from first inward fit kTPCin)
  - ***GetOuterParam()*** = Track parameters at the point of max. radial coordinate reached in the PropagateBack step
  - ***GetOuterHmpParam()*** = Track parameters at HMPID

FROM ESD TO AOD

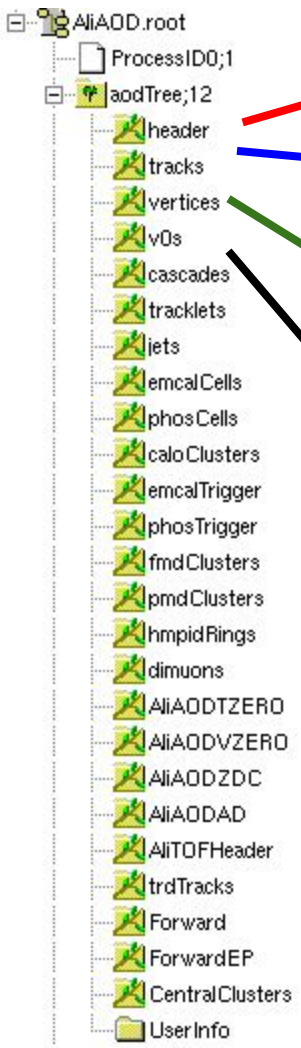
DPG ALICE • Data Preparation Group

Analysis Object Data (AOD)

- **AOD files** should be the main source for physics analysis
  - Lighter than ESDs
- Produced from ESDs as <u>last step of the PPass</u> (and muon_calo) reconstruction or in <u>dedicated AOD refiltering</u> productions
  - Analysis train for AOD creation defined in the macros ***AliDPG/AOD/AODtrain.C*** (raw data) and ***AliDPG/AOD/AODtrainsim.C*** (MC)
- The main "wagon" is the analysis task ***$ALICE_ROOT/ANALYSIS/ESDfilter/AliAnalysisTaskESDfilter.cxx***
  - Creates **AliAOD.root** and **AliAOD.Muons.root** files
  - In case of MC productions: the "kinematic tree" of generated particles is converted into a tree of AliAODMCParticle objects and stored in AliAOD.root
  - Additional information, e.g. for D-meson and photon coversion stored in dedicated delta-AOD files: **AliAOD.VertexingHF.root** and **AliAODGammaConversion.root**

AliAOD.root

- Similar structure as AliESDs.root



AliAODHeader: event properties

- **run number, mag field, diamond info ...**

Array of AliAODTracks

- **Central barrel tracks, muon tracks, kinks**

Array of vertices (primary, pileup, decay…)

- **AliAODVertex objects**

ARRAYS OF V0s, CASCADES

- **AliAODv0, AliAODcascade objects**

AliAODTrack

- Inherits from **AliVTrack** and not from AliExternalTrackParam)
- Different parameterisation w.r.t. AliESDtrack:

```
// Momentum & position
Double32_t    fMomentum[3];       // momemtum stored in pt, phi, theta
Double32_t    fPosition[3];       // position of first point on track or dca
```

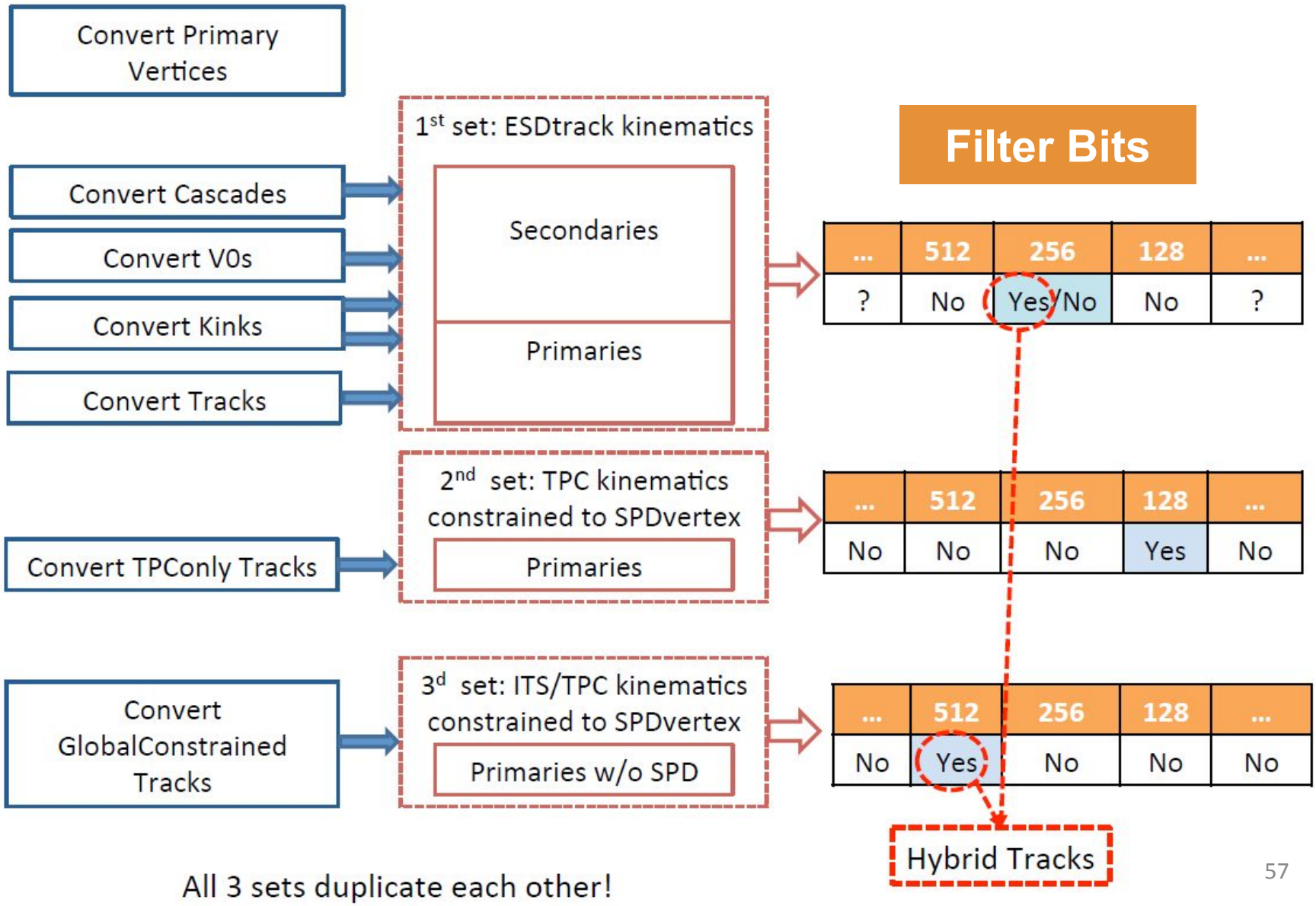
  - Momentum components and position at DCA to vertex
  - Covariance matrix
  - Only parameters at the DCA to the vertex are stored
- Can be converted to AlIExternalTrackParam via AliExternalTrackParam::CopyFromVTrack
- AOD tracks have a **filter-bit mask**
  - Stores the information about whether the track satisfies standard sets of quality criteria
  - Each filter bit correspond to a given set of cuts

DPG
ALICE • Data Preparation Group

Filter-bit example (AOD145)

Bit	Cuts	Methods
Bit 0 (001)	Standard cuts on primary tracks	GetStandardTPCOnlyTrackCuts() (*)
Bit 1 (002)	ITS stand-alone tracks(ESD Track Cuts)	SetRequireITSStandAlone(kTRUE)
Bit 2 (004)	Pixel OR (necessary for the electrons) AND Standard track cuts (SetFilterMask(1) of AliESDtrackCuts)	SetClusterRequirementITS(AliESDtrackCuts::kSPD, AliESDtrackCuts::kAny)
Bit 3 (008)	PID for the electrons AND Pixel Cuts (SetFilterMask(4) of AliESDpidCuts)	SetTPCnSigmaCut(AliPID::kElectron, 3.5)
Bit 4 (016)	Standard Cuts with very loose DCA	GetStandardITSTPCTrackCuts2011(kFALSE) SetMaxDCAToVertexXY(2.4) SetMaxDCAToVertexZ(3.2) SetDCaToVertex2D(kTRUE)
Bit 5 (032)	Standard Cuts with tight DCA cut	GetStandardITSTPCTrackCuts2011()
Bit 6 (064)	Standard Cuts with tight DCA but with requiring the first SDD cluster instead of an SPD cluster tracks selected by this cut are exclusive to those selected by the previous cut	GetStandardITSTPCTrackCuts2011() SetClusterRequirementITS(AliESDtrackCuts::kSPD,AliESDtrackCuts::kNone) SetClusterRequirementITS(AliESDtrackCuts::kSDD,AliESDtrackCuts::kFirst)
Bit 7 (128)	TPC only tracks,constrained to SPD vertex in the filter	GetStandardTPCOnlyTrackCuts **esdfilter->SetTPCOnlyFilterMask(128)**
Bit 8 (256)	Extra cuts for Hybrids: - first the global tracks we want to take	AliESDtrackCuts::GetStandardITSTPCTrackCuts2011(kFALSE) SetMaxDCAToVertexXY(2.4) SetMaxDCAToVertexZ(3.2) SetDCAToVertex2D(kTRUE) SetMaxChi2TPCConstrainedGlobal(36) SetMaxFractionSharedTPCClusters(0.4) **esdfilter->SetHybridFilterMaskGlobalConstrainedGlobal((1<<8));** // these normal global tracks will be marked as hybrid
Bit 9 (512)	Than the complementary tracks which will be stored as global constraint, complement is done in the ESDFilter task	SetClusterRequirementITS(AliESDtrackCuts::kSPD,AliESDtrackCuts::kOff) SetRequireITSRefit(kTRUE) **esdfilter->SetGlobalConstrainedFilterMask(1<<9);** // these tracks are written out as global constrained tracks esdfilter->SetWriteHybridGlobalConstrainedOnly(kTRUE); // write only the complement
Bit 10(1024)	Standard Cuts with tight DCA cut, using cluster cut instead of crossed rows	GetStandardITSTPCTrackCuts2011(kTRUE,0) (**)

ESD -> AOD filtering



Global and TPC only tracks

- **Global tracks**
  - Treated in AliAnalysisTaskESDfilter::ConvertTracks
  - An ESD track is converted into AOD track and stored in the AOD track array if:
    - it passes at least 1 of the selection (filter bit) cuts
    - OR if it belongs to vertex contributors (e.g. was used for primary vertex reconstruction)

- **TPC only tracks**: filter bit 128 (1<<7) - all other bits are reset
  - Treated in AliAnalysisTaskESDfilter::ConvertTPCOnlyTracks
  - All tracks passing <u>standard TPC-only cuts</u> stored with:
    - TPC-only track parameters from kTPCin step (*GetTPCInnerParam*) constrained to SPD vertex
    - Unconstrained momentum and position are also stored
    - AOD track ID, *AliAODTrack::GetID()*, set to *negative value*

Hybrid tracks

- **Global hybrid tracks**: filter bit 256 (1<<8)
  - Treated in AliAnalysisTaskESDfilter::ConvertTracks
  - Tracks passing standard TPC+ITS cuts , SPD hit request and golden chi2 cut are marked as filter bit 256
- **Complementary hybrid tracks**: filter bit 512 (1<<9) - all other bits are reset
  - AliAnalysisTaskESDfilter::ConvertGlobalConstrainedTracks
  - All tracks passing standard TPC+ITS cuts have no hits in SPD are stored in the AOD track array with
    - Global track parameters constrained to primary vertex (*GetConstrainedParam*)
    - Unconstrained momentum and position are also stored
    - AOD track ID, *AliAODTrack::GetID()*, set to *negative value*

EVENT SELECTION

Triggers in ALICE

- The ALICE experiment cannot record all events:
  - *Large detector readout time (up to 1 ms/event)*
  - *Data storage limitations*
- We need some non-negligible activity in ALICE detectors to start the data acquisition (**Minimum bias triggers**)
  - Mainly based on V0, SPD and T0:
    - pp 2010: SPD | V0A | V0C (**INT1**)
    - pp 2012: T0A & T0C within a given time window (**INT8**)
    - pp 2015-2017 and p-Pb: V0A & V0c (**INT7**)
- We may be interested in rare observables for which we can collect enough statistics only requiring specific signatures in ALICE the detectors (**rare triggers**)
  - Most common rare triggers (used in coincidence with SPD or V0 or T0 triggers)
    - Muon triggers (single muon, dimuon)
    - Calorimeter triggers (EMCAL, PHOS)
    - Ultra-peripheral collisions (Muon, T0F+SPD)

Triggers classes

Trigger Classes: product of several trigger requirements

- *Typical trigger class name:* **CEMC7EGA-B-NOPF-CENTNOTRD**

Descriptor — BC mask — PF protection — Cluster

Descriptor: *combination of trigger inputs (AND, OR, VETO logic is possible)*
BC mask: *information about interactions (beam-beam, beam-gas, satellite, etc.)*
Past-Future protection: *rejects events with multiple collisions from different bunch crossings*
Cluster: *group of detectors to be read out if the trigger conditions is satisfied*

CLUSTERS	Detectors																			
	ACORDE	CPV	DAQ TEST	EMCAL	FMD	HMPID	MUON TRG	MUON TRK	PHOS	PMD	SDD	SPD	SSD	T0	TOF	TPC	TRD	TRIGGER	V0	ZDC
ALLNOTRD	✓			✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CENTNOTRD	✓			✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓		✓	✓	✓
ALL	✓			✓	✓	✓					✓	✓	✓	✓	✓	✓		✓	✓	✓
CENT	✓			✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FAST	✓			✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MUON							✓	✓				✓		✓				✓	✓	✓

Trigger aliases

➔ Definitions of trigger aliases in: `AliVEvent::EOfflineTriggerTypes`

Trigger	Definition
AliVEvent::kMB	BIT(0), Min. bias trigger in PbPb 2010-11
AliVEvent::kINT1	BIT(0), V0A | V0C | SPD minimum bias trigger
AliVEvent::kINT7	BIT(1), V0AND minimum bias trigger
AliVEvent::kMUON	BIT(2), Single muon trigger in pp2010-11, INT1 suite
AliVEvent::kHighMult	BIT(3), High-multiplicity SPD trigger
...	...
AliVEvent::kAny	0xffffffff, to accept any defined trigger

(See all trigger definitions in `$ALICE_ROOT/STEER/STEERBase/AliVEvent.h`)

➔ Assignment of trigger classes to aliases and definition of background and quality checks is stored in OADB. Check details in `$ALICE_PHYSICS/OADB/macros/BrowseAndFillPhysicsSelectionOADB.C`

➔ Several aliases can be fired at the same time

Physics Selection

The Physics Selection class (`AliPhysicsSelection`) is used to select collision candidates in data samples collected by ALICE

- Reject background and poor quality events according to predefined requirements
- Select events within the trigger class fired
- Aliases can be used to group similar trigger class names:

  Example: alias `AliVEvent::kMUSH7`

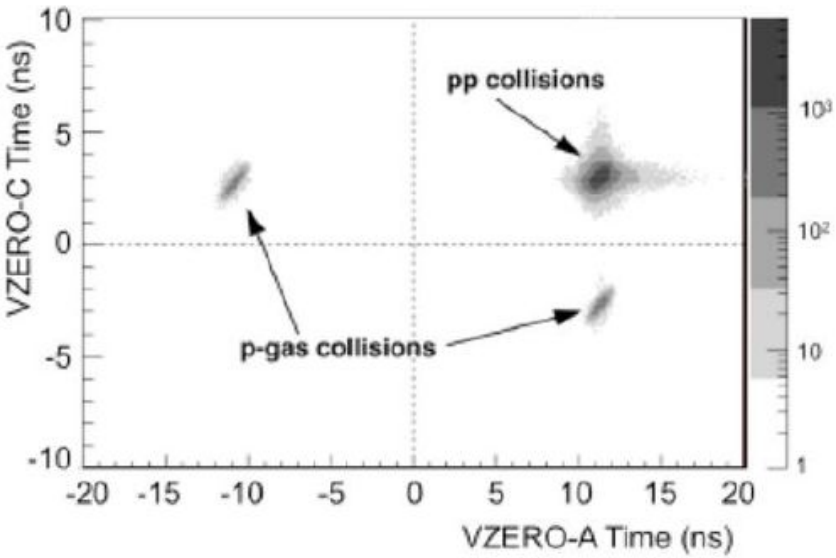
  - CMSH7-B-NOPF-MUON
  - CMSH7-S-NOPF-MUON
  - CMSH7-S-NOPF-ALLNOTRD
- Basic usage: *how to select the events in your analysis task*

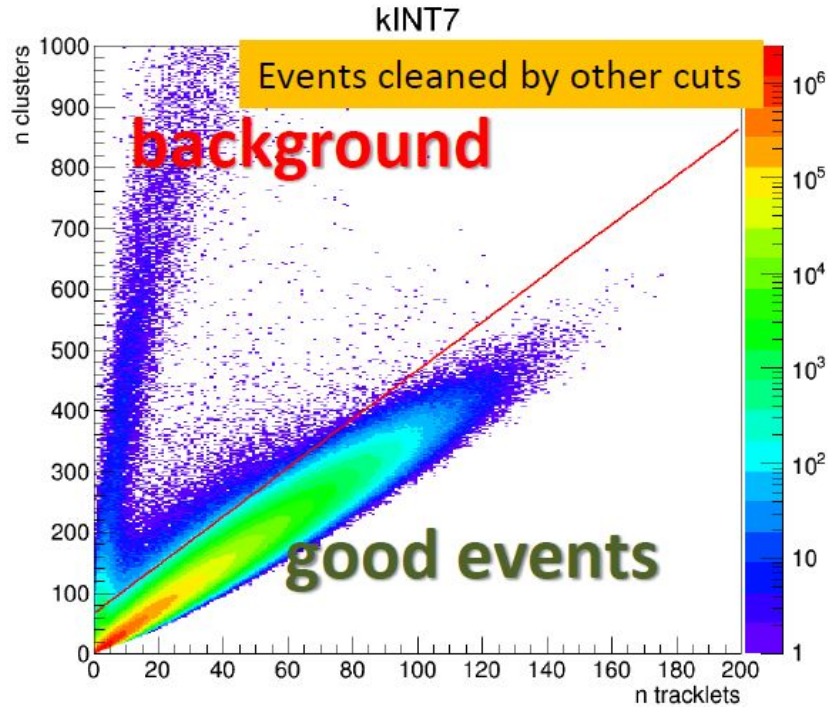
```
// in your UserExec
UInt_t fSelectMask = fInputHandler->IsEventSelected();
Bool_t isINT7selected = fSelectMask & AliVEvent::kINT7;
```

(see more in https://twiki.cern.ch/twiki/bin/view/ALICE/AliDPGtoolsEventProp#Event_selection)

Physics Selection

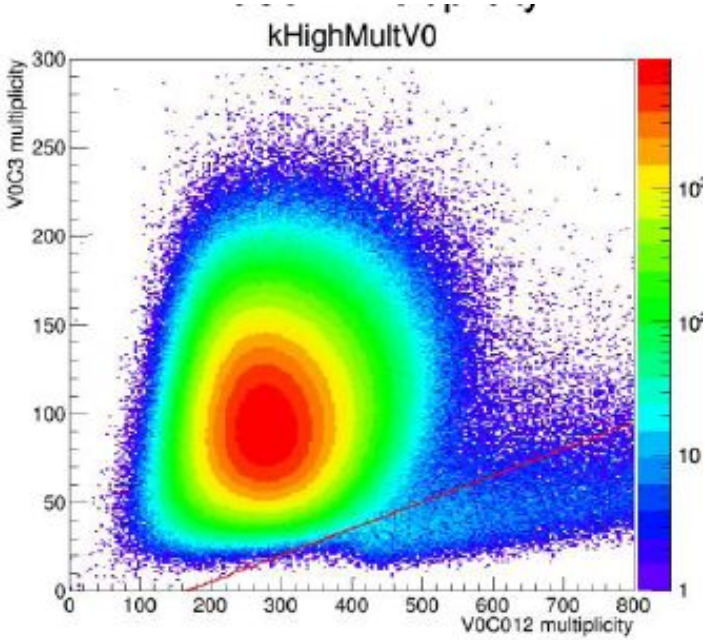
- Basic selections:
  - V0A and V0C time information
  - Cluster-vs-tracklet background cut (now in user hands)
  - ZDC timing cuts (in Pb-Pb and p-Pb)
  - Incomplete event rejection (in p-Pb, pp2015)

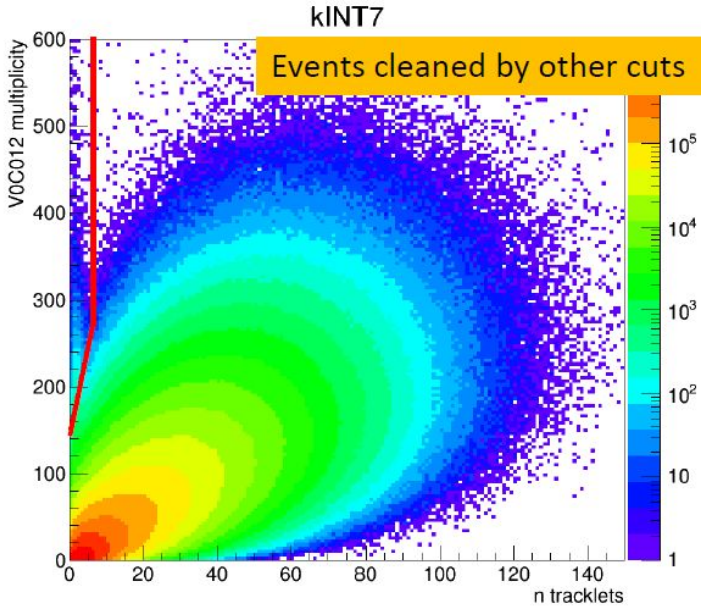




Physics Selection

- Basic selections:
  - V0A and V0C time information
  - Cluster-vs-tracklet background cut (now in user hands)
  - ZDC timing cuts (in Pb-Pb and p-Pb)
  - Incomplete event rejection (in p-Pb, pp2015)
- New background cuts
  - V0C012 vs V0C3 asymmetry cut
  - V0C012 vs tracklet background cut





Physics Selection

- Basic selections:
  - V0A and V0C time information
  - Cluster-vs-tracklet background cut (now in user hands)
  - ZDC timing cuts (in Pb-Pb and p-Pb)
  - Incomplete event rejection (in p-Pb, pp2015)
- New background cuts
  - V0C012 vs V0C3 asymmetry cut
  - V0C012 vs tracklet background cut
- Pileup cuts
  - Out-of-bunch pileup cut based on V0 past-future info
  - Out-of-bunch pileup cut based on online-vs-offline V0M correlation
  - Out-of-bunch pileup cut based on online-vs-offline SPD FO correlation

Pileup removal

Two main categories of pileup:

1. Same bunch-crossing pileup

(two or more collisions occurring in the same bunch crossing)

- ➔ seen by all detectors
- ➔ reconstructed points in drift detectors (TPC and SDD) are in the "correct" spatial position
- ➔ can be removed at the ***event selection*** level with cuts based on ***multiple reconstructed vertices***

2. Out-of-bunch pileup

(one or more collisions occurring in bunch crossings different from the one which triggered the acquisition)

- ➔ detectors are affected differently depending on their readout time
- ➔ reconstructed points in drift detectors are spatially shifted (z for TPC, rϕ for SDD) by $\Delta s = v^{drift} \cdot \Delta t$
- ➔ can be removed at the ***event selection*** level and also based on ***track selection*** cuts

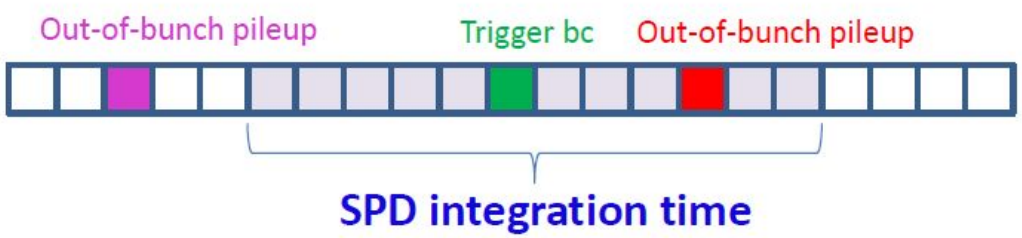
Out of bunch pileup

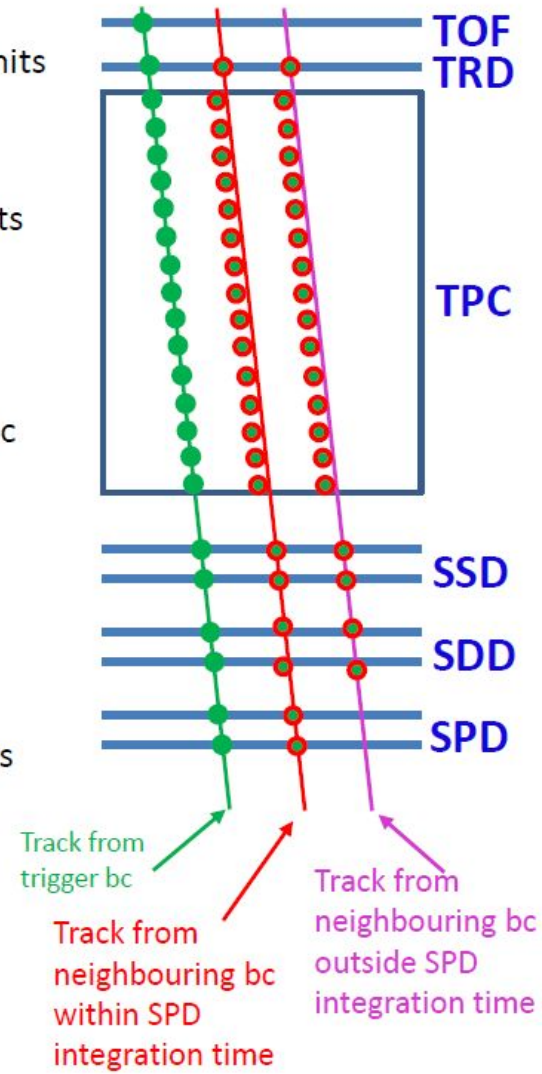
Integration times in central barrel:

- **SPD**: 300 ns (12 bcs) - out-of-bunch hits indistinguishable from trigger hits
- **SDD:** ~6 us (240 bcs) – radial drift ~0.5mm/100ns.
  Tracking tolerance: σ~0.5 mm for pp (0.2 mm for Pb-Pb)
- **SSD:** ~1 us (40 bcs) - out-of-bunch hits indistinguishable from trigger hits
- **TPC:** ~100 us (4000 bcs) – z-drift ~2.5mm/100ns.
  ITS-TPC track matching tolerance: σ~5 mm
- **TRD:** ~1 us (40 bcs) - radial drift ~2.5mm/100ns.
- **TOF**: ~0.5 us (20 bcs) – time info allows to identify tracks from trigger bc however not all analyses require TOF hit matching

Conclusions:

- SPD hit requirement cleans up tracks from neighbouring bunches
  - Does not help in case of pileup within SPD integration time
- Need to remove residual out-of-bunch pileup in SPD integration time
- Out-of-bunch pileup removal is crucial in multiplicity-differential studies



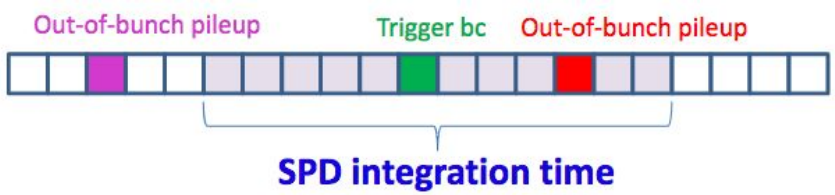


Pileup removal

Tools for pileup tagging, removal, mitigation:

(see dedicated Twiki page for details: https://twiki.cern.ch/twiki/bin/view/ALICE/AliDPGtoolsPileup)

- **Past-Future protection:** allows to remove out-of-bunch pileup from outside the SPD readout time (300ns) - *do not help within SPD integration time window*



Should be activated for pp and p-Pb analyses in the Physics Selection task:

```
AddTaskPhysicsSelection(isMC, kTRUE)
```

- **Multiple vertices with the SPD:** sensitive to same-bunch and out-of-bunch pileup within the SPD readout window
  → *can be accessed via:*

```
AliESDEvent::IsPileUpFromSPD()
AliESDEvent::IsPileUpFromSPDInMultBins() //mult. dep.
```

- **Multiple vertices with tracks:** simultaneous vertex finding using ITS, TPC and TOF (if available) - *wider time coverage than SPD*
  → *can be accessed via:*

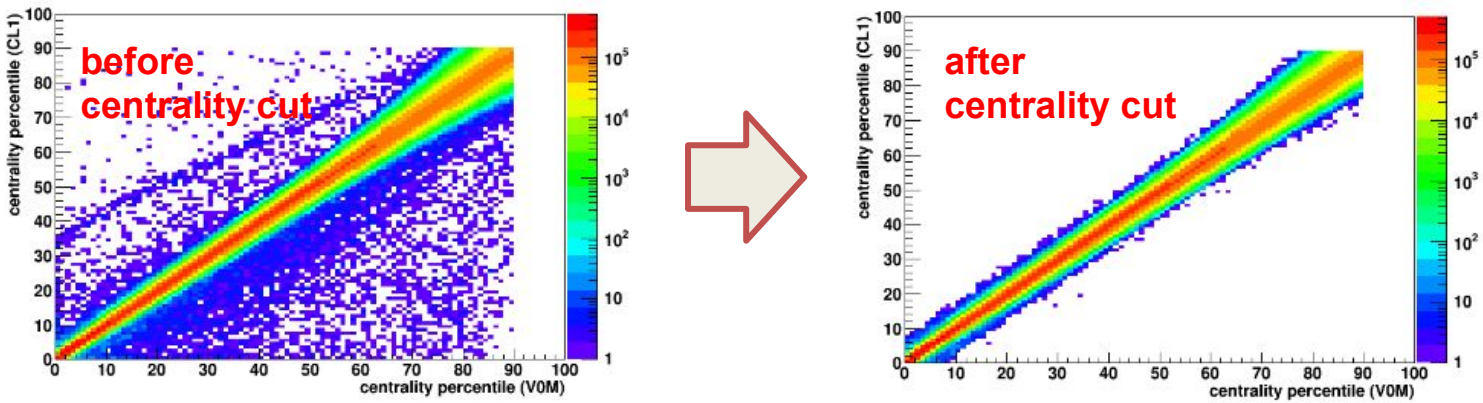
```
AliAnalysisUtils::IsPileUpMV(AliVEvent *event)
```

Pileup removal

Tools for pileup tagging, removal, mitigation:

(see dedicated Twiki page for details: https://twiki.cern.ch/twiki/bin/view/ALICE/AliDPGtoolsPileup)

- **Correlations between centrality estimators (Pb-Pb):** cut on correlation between V0 and CL0, CL1 centralities



- **Removing tracks from out-of-bunch pileup collisions:**
  - *require tracks with **ITS hits** (in particular to SPD)*
  - *require matching to TOF and **TOF bunch crossing ID=0***
  - *require tracks to point to the main vertex via **DCAz cut***

Primary Vertex selection

Primary vertex information can be retrieved from ESD and AOD events

```
AliVVertex* vtx = event->GetPrimaryVertex();
```

(AliVVertex can be casted to ESD or AOD vertex)

This method will return (following this order):

1. *the vertex reconstructed from tracks*
2. *the SPD vertex (if the track vertex is not available)*
3. *the vertex from TPC tracks (if also the SPD vertex is not available)*

A set of selections can be applied at the analysis level in order to reject events with poorly reconstructed vertex

- ➔ *Selection on **contributors to vertex***
- ➔ *Selection on SPD **vertex type** (3D or z reconstruction)*
- ➔ *Special selections for SPD vertex (based on z resolution/dispersion)*
- ➔ *Cut on **absolute distance between track and SPD vertices** (Pb-Pb 2011)*
- ➔ *Cut on **absolute and nsigma distance between track and SPD vertices** (Pb-Pb 2015)*

By construction, the main vertex is the one with the largest number of contributors - *additional vertices found are treated as **pileup** vertices*

Event Selection Class: AliEventCuts

A general class for event selections is available:

(see more instructions in: https://twiki.cern.ch/twiki/bin/view/ALICE/AliDPGtoolsEventProp#Event_selection_class)

- ➔ main functionalities similar to those of `AliESDtrackCuts`
- ➔ provides a method to check if a given AliVEvent is accepted: `AliEventCuts::AcceptEvent(AliVEvent *ev)`
- ➔ Simple usage:

Add to the class member of your task:

```
class MyAliAnalysisTask {
  public:
  ...
  AliEventCuts fEventCuts; // Event cuts
  ...
}
```

Enable selection QA plots (if wanted):

```
// fList is your output TList
fEventCuts.AddQAplotsToList(fList);
```

In your `UserExec` *method:*

```
AliVEvent *ev = InputEvent();
if (!fEventCuts.AcceptEvent(ev)) {
  PostData(1, fList);
  return;
}
```

Event Selection Class: AliEventCuts

A general class for event selections is available:

(see more instructions in: https://twiki.cern.ch/twiki/bin/view/ALICE/AliDPGtoolsEventProp#Event_selection_class)

➔ Retrieves the information about primary vertex and multiplicity after calling `AliEventCuts::AcceptEvent(AliVEvent* ev)`

```
float centrality = fEventCuts.GetCentrality(); //centrality from the default estimator
const AliVVertex* vtx = fEventCuts.GetPrimaryVertex(); //best primary vertex available
```

➔ Advanced usage: *set cuts manually*

```
fEventCuts.SetManualMode(); //Enable manual mode
fEventCuts.fRequireTrackVertex = true;
fEventCuts.fMinVtz = -10.f;
fEventCuts.fMaxVtz = 10.f;
fEventCuts.fMaxDeltaSpdTrackAbsolute = 0.5f;
fEventCuts.fMaxResolutionSPDvertex = 0.25f;
fEventCuts.fTriggerMask = AliVEvent::kINT7;
fEventCuts.fRejectDAQincomplete = true;
fEventCuts.fSPDpileupMinContributors = 3;
fEventCuts.fSPDpileupMinZdist = 0.8;
fEventCuts.fSPDpileupNsigmaZdist = 3.;
fEventCuts.fSPDpileupNsigmaDiamXY = 2.;
fEventCuts.fSPDpileupNsigmaDiamZ = 5.;
fEventCuts.fTrackletBGcut = true;
```

example of manual settings used in pp Run 2 event selection

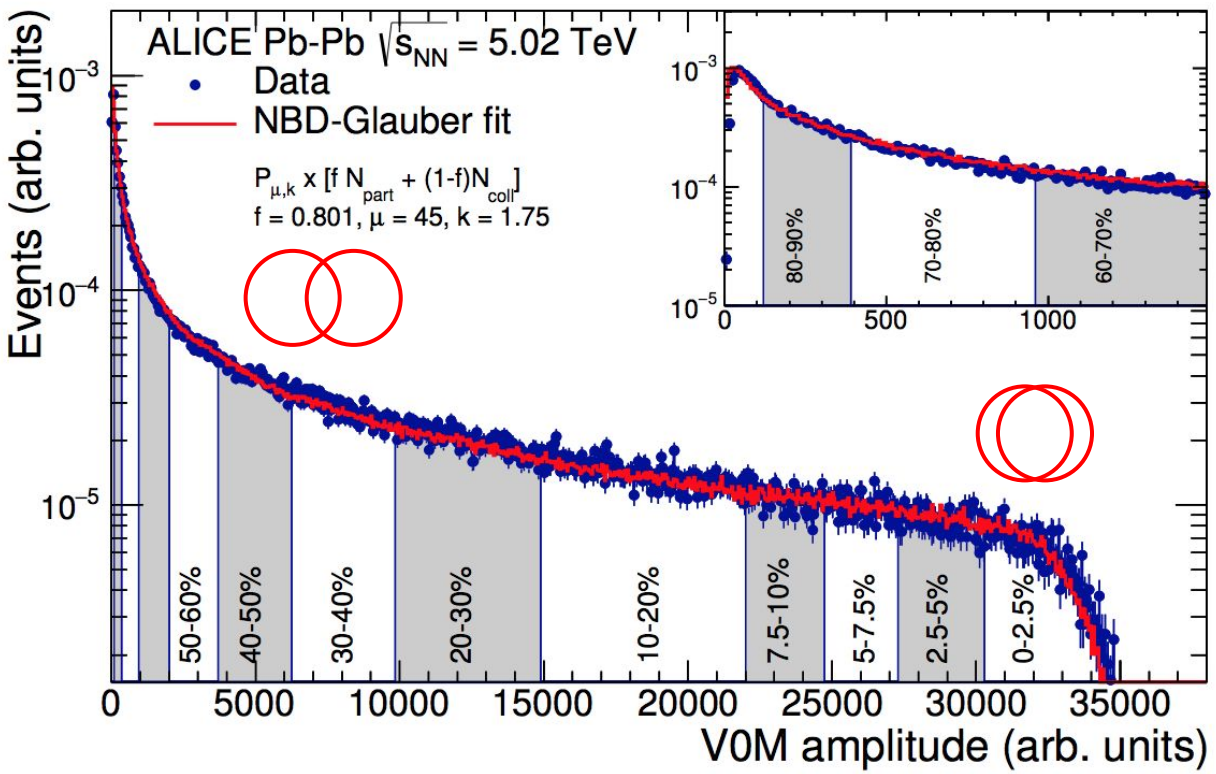
(add in your AddTask or in the UserCreateOutputObjects method)

EVENT CHARACTERISATION

More info here:
https://indico.cern.ch/event/387210/contributions/918223/attachments/1203723/1753174/DDChinellato-AliMultSelection-PhysicsForum1.pdf

Centrality selection in Pb-Pb

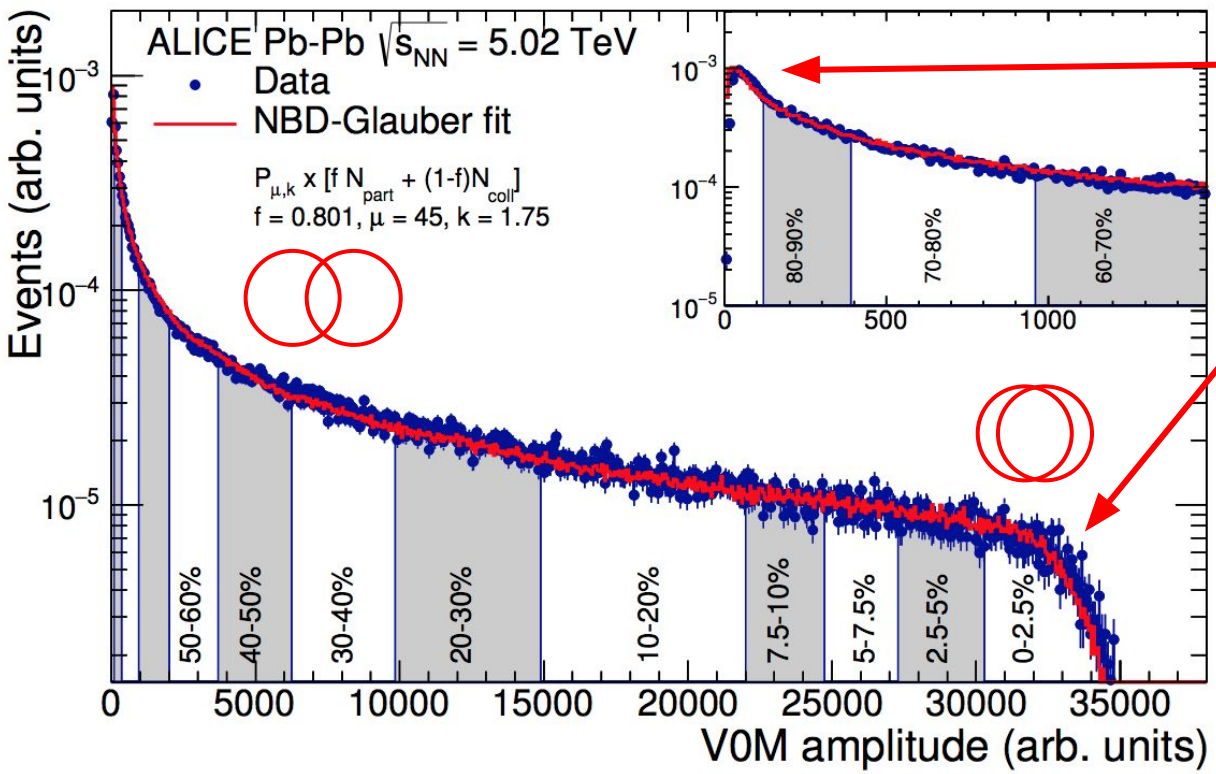
- Selecting events by impact parameter: impossible
- Next best thing: **select events** based on charged particle multiplicity measured in the V0 detectors: forward **η** to minimize auto-correlation biases
- Done in the "**AliMultSelection**" framework (also in pp, p-Pb!)



more info: https://aliceinfo.cern.ch/Notes/node/456

Centrality selection in Pb-Pb

- Selecting events by impact parameter: impossible
- Next best thing: **select events** based on charged particle multiplicity measured in the V0 detectors: forward **η** to minimize auto-correlation biases
- Done in the "**AliMultSelection**" framework (also in pp, p-Pb!)



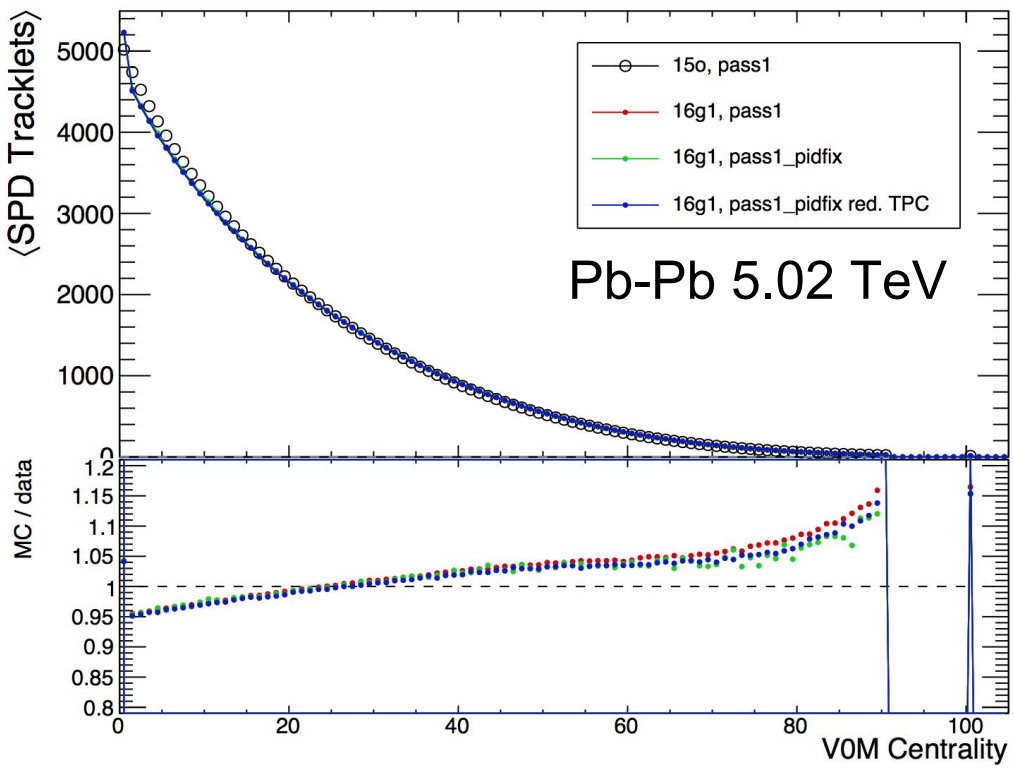
Very low multiplicity: discarded due to contamination from EM processes

Glauber MC model: A geometric approach to calculate participating and colliding nucleons **In this plot**: coupled to a negative binominal (NBD) for describing multiplicity

more info: https://aliceinfo.cern.ch/Notes/node/456

Centrality in Pb-Pb: MC

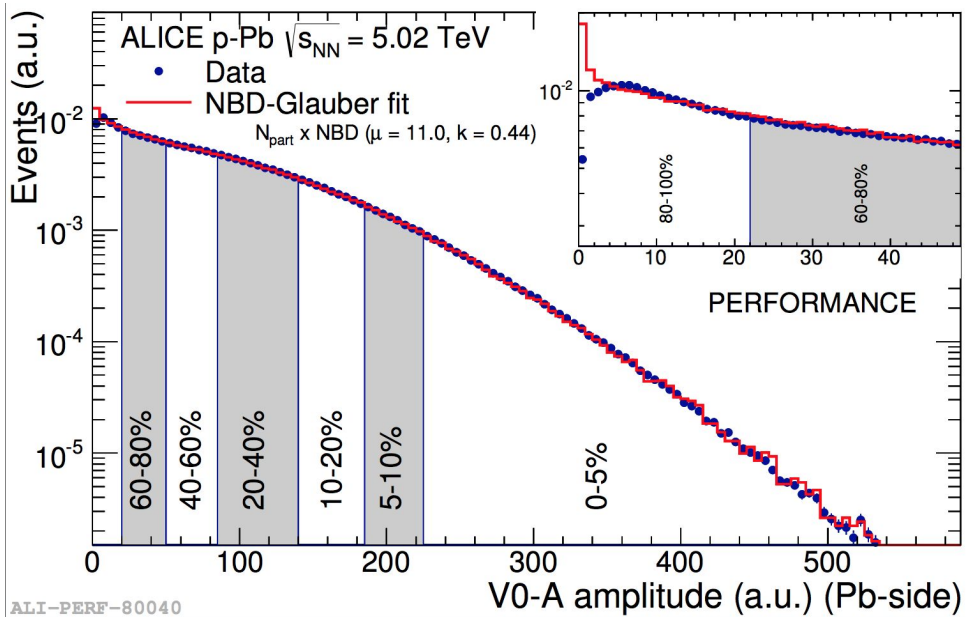
- In MC, the objective of the centrality selection is not to produce *percentiles* but rather to provide a $<N_{ch}>$ for *a given percentile selection* that matches what is seen in data: ideal for **efficiency correction calculations**

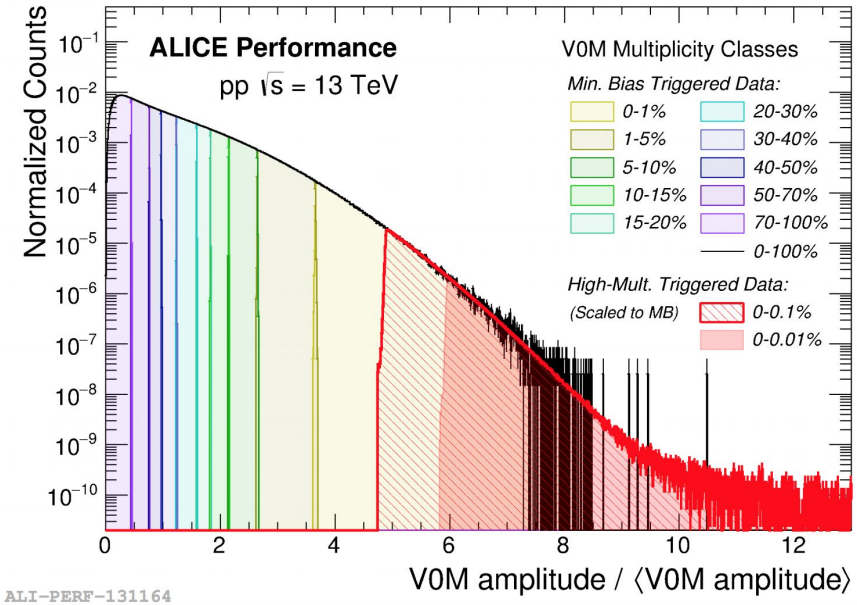


- Works fairly well up to a level of approximately ~5%-10%
- Strongly centrality dependent corrections usually assigned a small systematic uncertainty due to this
- The <SPD tracklet> plot is **available as standard QA** in AliMultSelection for x-checks
  - (more info here)

more info: https://aliceinfo.cern.ch/Notes/node/456

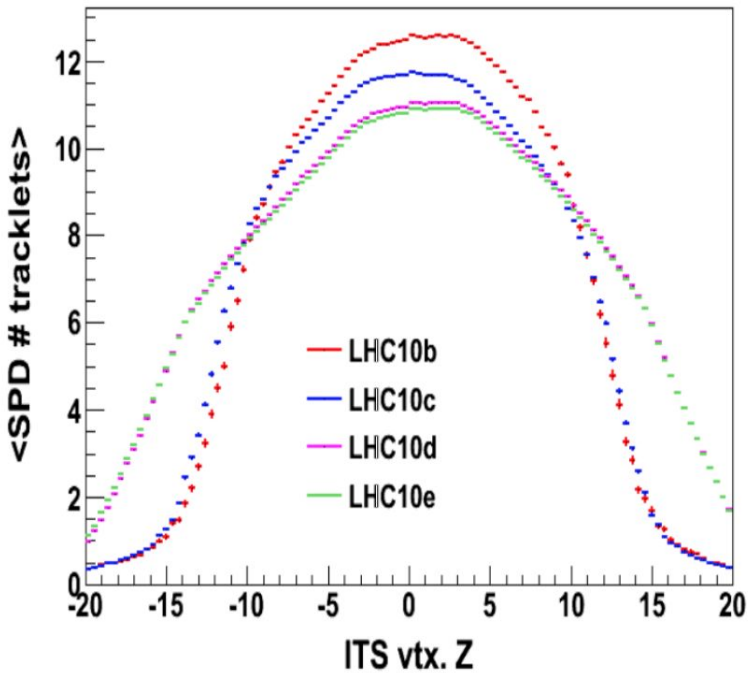
Centrality in p-Pb, pp





- pp and p-Pb are also sliced in V0A+V0C amplitudes, with notable differences:
- **pp**: typically we select slices of the INEL>0 cross-section, with the INEL>0 condition being "at least one SPD tracklet inside |η|<1.0". **Correcting to the physical INEL>0 event class** has to be done at analysis level
- **p-Pb**: typically we select slices of the visible V0A cross-section. **Correcting for the vertex determination efficiency** has to be done at analysis level
- (N.B.: corrections not needed in Pb-Pb: close to 100% efficiency!)

Centrality: vertex-Z corrections

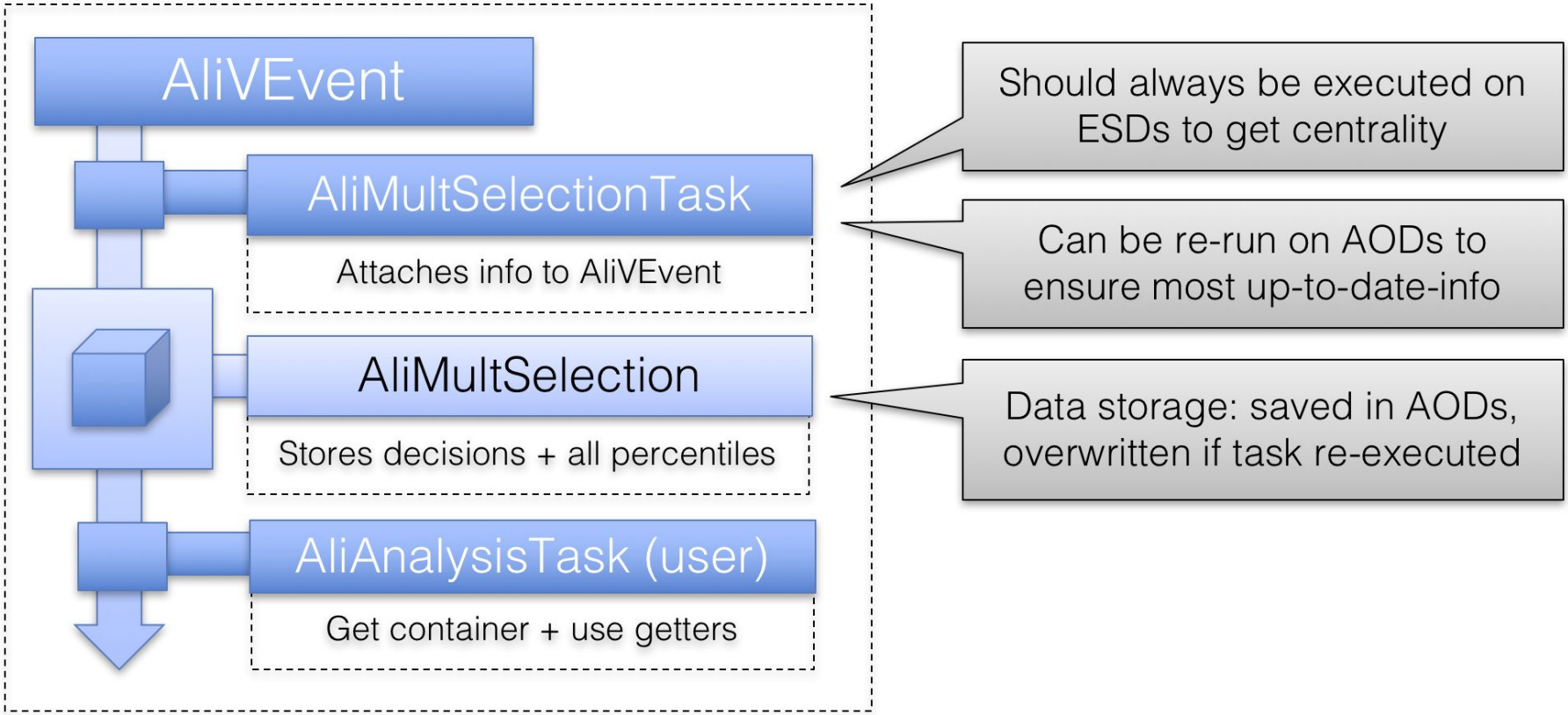


- Our **detector acceptance changes** depending on the PV position along the beam axis (z direction)
- Thus, the average raw value of any estimator may change with vertex-Z
- **This can be corrected for**: instead of calibrating based on N_{raw}, we calibrate N_{corr}:

$$N_{corr} = N_{raw} / \langle N_{raw}\ (\text{vtx-Z})\rangle$$

- $\langle N_{raw}\ (\text{vtx-Z})\rangle$ is usually a polynomial fit to the averages measured in data. **Example in figure: pp @ 7TeV**
- May be more or less important (V0M -> dependence partially cancels out as V0A↑ = V0C↓ and vice-versa, SPD-based estimators -> very sensitive)

AliMultSelection: operation



Analysis usage consists of:

1) Get AliMultSelection object

```
AliMultSelection *obj = (AliMultSelection*) lVEvent-> FindListObject("MultSelection");
```

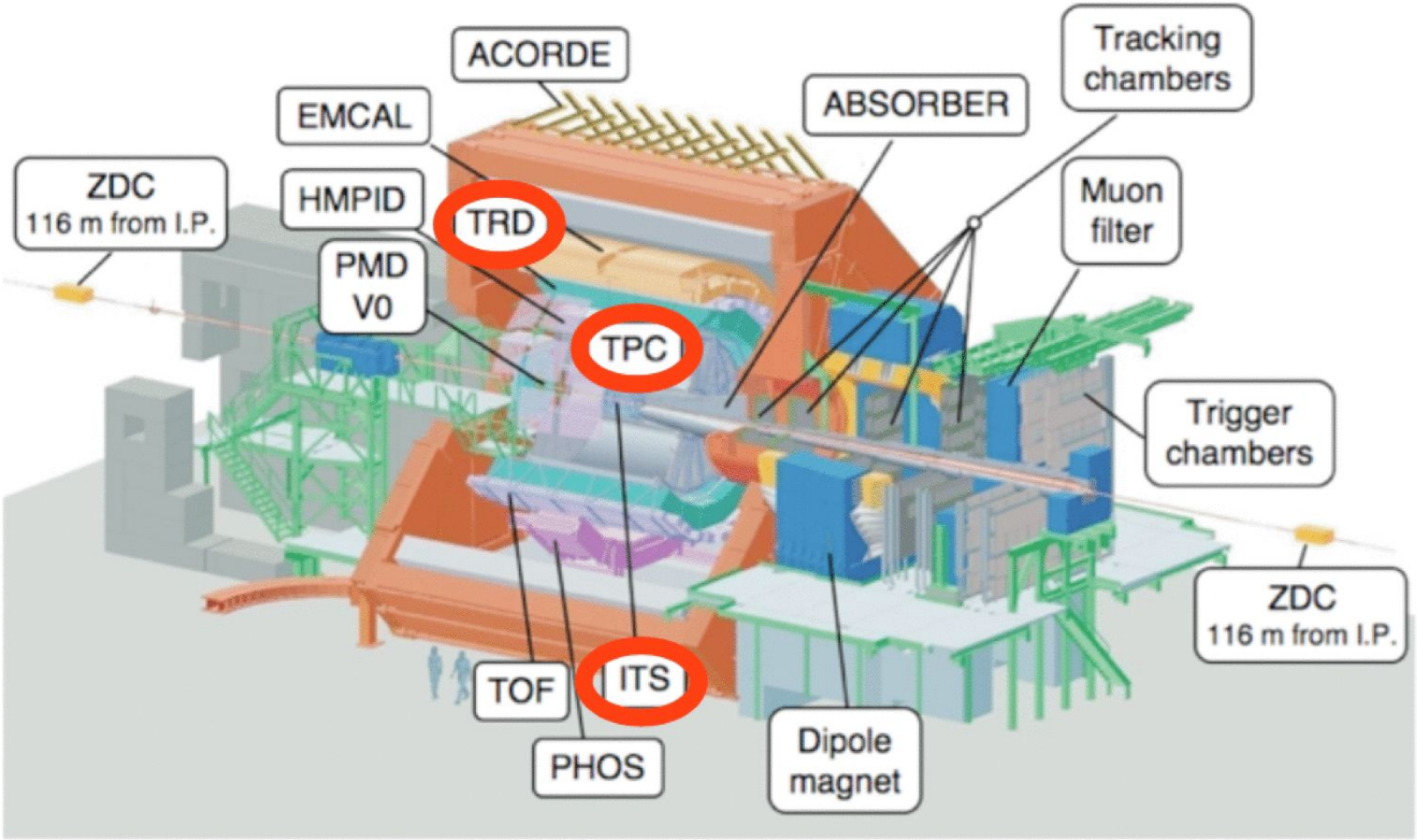
2) Get desired multiplicity percentile

```
obj->GetMultiplicityPercentile("V0M");
```

TRACK SELECTION on ESD and AOD

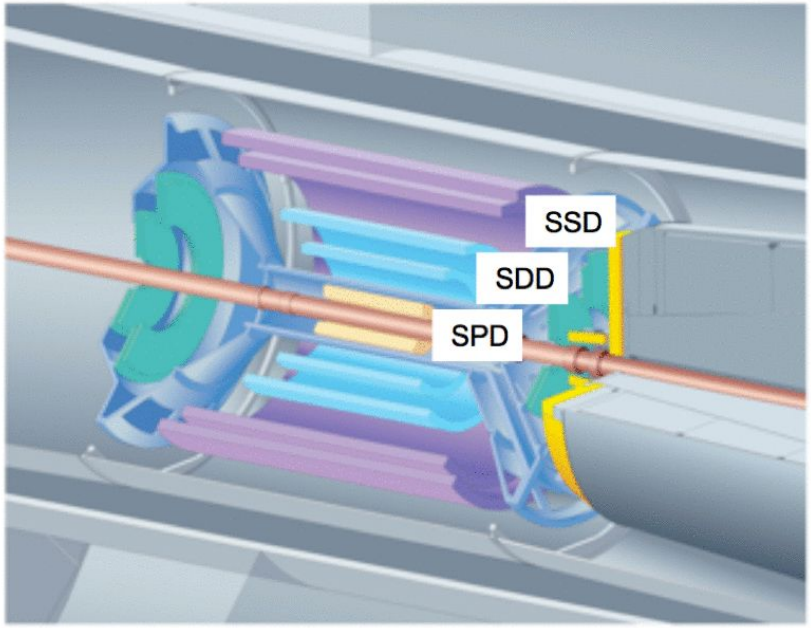
(also using material from the tutorial prepared by Z Conesa del Valle and A Kalweit for the ALICE Physics Week 2013)

ALICE Barrel tracking detectors



Inner Tracking System

- 3 sub-detectors with in total 6 layers
- Silicon Pixel detector: SPD
- Silicon Drift detector: SDD
- Silicon Strip detector: SSD



- Up to 6 hits (clusters) in the ITS can be attached to a track:

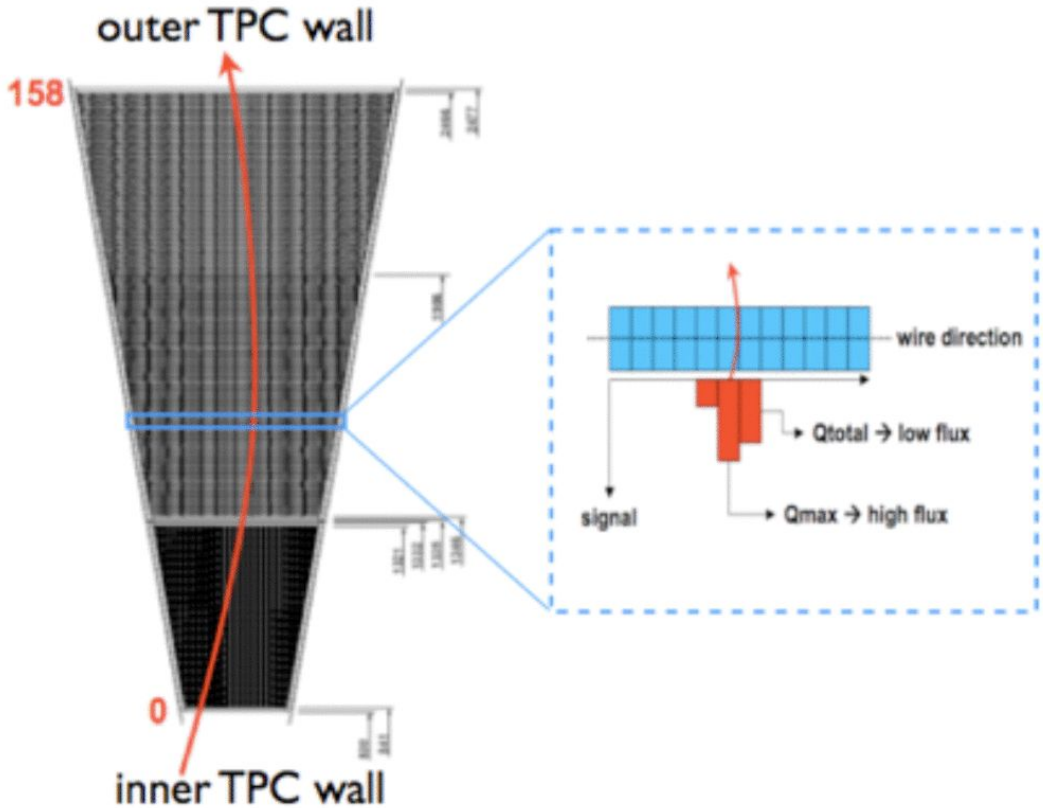
AliESDtrack::GetNcls(Int_t idet = 0)
AliESDtrack::HasPointOnITSLayer(Int_t i)

AliAODTrack::GetITSNcls()
AliAODTrack:: HasPointOnITSLayer(Int_t i)

Time Projection Chamber

- The ALICE TPC provides up to 159 space points (clusters) corresponding to the number of pad rows.

- The signal can be below threshold for low ionizing particles.



TPC clusters

- The ALICE TPC provides up to 159 space points (clusters) corresponding to the number of pad rows.

- The signal can be below threshold for low ionizing particles.

Def. 1 (*TPC cluster*)

A charged particle traversing the TPC induces a signal on a given pad-row. If the charge in a search window of 5 pads in wire direction and 5 bins in time direction exceeds a certain threshold and fulfills all necessary quality criteria, it is called a cluster. Therefore the maximum number of clusters per track n_{cl} is 159, which corresponds to the number of pad rows in a TPC sector. Curling track parts are reconstructed as separate tracks. The number of clusters assigned to a track is related to the track length in the sense that low p_t-tracks which do not reach the outer wall of the TPC have less clusters assigned. However, the relation is not straightforward, because the pad length in the TPC is increasing with radial distance to the center.

DPG ALICE • Data Preparation Group

ESD track selection: AliESDtrackCuts

- There are default cut sets available which are a very good starting point for all analyses (however, there is not a default cut set which is perfect for every analysis):

```
AliESDtrackCuts * fESDTrackCuts = AliESDtrackCuts::GetStandardITSTPCTrackCuts2010(kTRUE);
fESDTrackCuts->SetEtaRange(-0.8,+0.8);
```

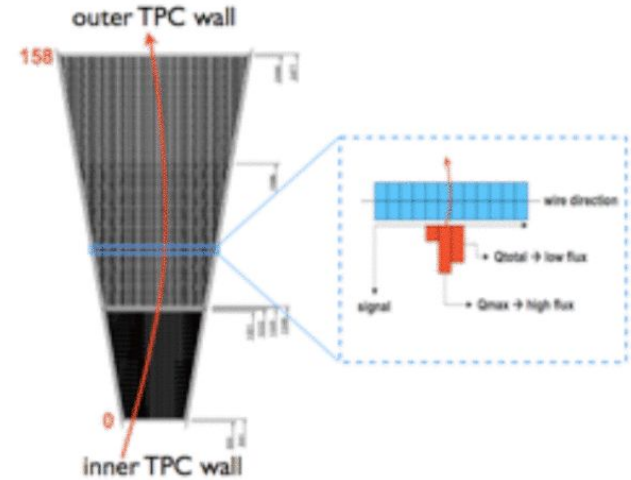
- Inside the track loop:

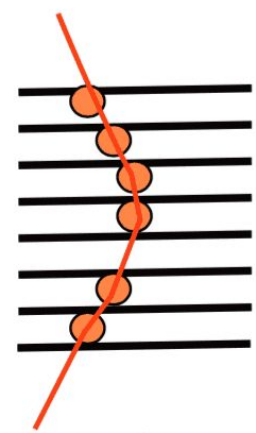
```
if (!fESDTrackCuts->AcceptTrack(track)) continue;
```

- Standard cut sets are defined for 2010 and 2011
- The 2011 standard cuts are a valid baseline also for Run2 data
- A new method for 2015 (and other Run2) data is in preparation, but will have essentially the same defaults cuts as the 2011 method

AliESDtracks: TPC cuts

```
AliESDtrackCuts* esdTrackCuts = new AliESDtrackCuts;
// TPC
if(clusterCut == 0)  esdTrackCuts->SetMinNClustersTPC(50);
else if (clusterCut == 1) {
  esdTrackCuts->SetMinNCrossedRowsTPC(70);
  esdTrackCuts->SetMinRatioCrossedRowsOverFindableClustersTPC(0.8);
}
else {

  esdTrackCuts->SetMinNClustersTPC(50);
}
esdTrackCuts->SetMaxChi2PerClusterTPC(4);
esdTrackCuts->SetAcceptKinkDaughters(kFALSE);
esdTrackCuts->SetRequireTPCRefit(kTRUE);
// ITS
esdTrackCuts->SetRequireITSRefit(kTRUE);
esdTrackCuts->SetClusterRequirementITS(AliESDtrackCuts::kSPD,
                                       AliESDtrackCuts::kAny);
if(selPrimaries) {
  // 7*(0.0015+0.0050/pt^1.1)
  esdTrackCuts->SetMaxDCAToVertexXYPtDep("0.0105+0.0350/pt^1.1");
  esdTrackCuts->SetMaxChi2TPCConstrainedGlobal(36);
}
esdTrackCuts->SetMaxDCAToVertexZ(2);
esdTrackCuts->SetDCAToVertex2D(kFALSE);
esdTrackCuts->SetRequireSigmaToVertex(kFALSE);

esdTrackCuts->SetMaxChi2PerClusterITS(36);
```

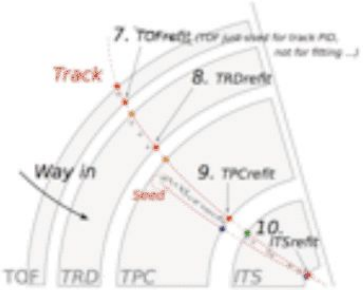




6 clusters / 7 crossed rows

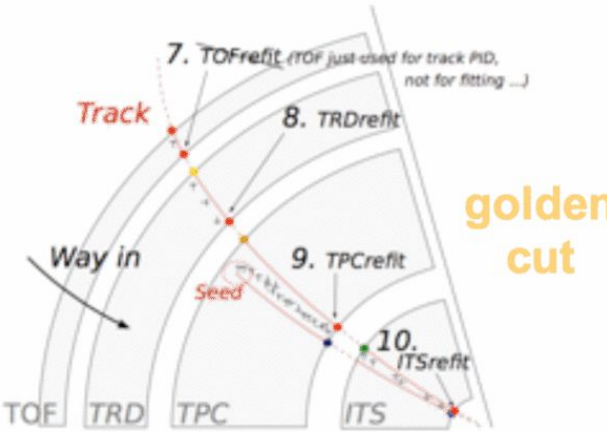
AliESDtrackCuts: TPC cuts

```
AliESDtrackCuts* esdTrackCuts = new AliESDtrackCuts;
// TPC
if(clusterCut == 0)  esdTrackCuts->SetMinNClustersTPC(50);
else if (clusterCut == 1) {
  esdTrackCuts->SetMinNCrossedRowsTPC(70);
  esdTrackCuts->SetMinRatioCrossedRowsOverFindableClustersTPC(0.8);
}
else {

  esdTrackCuts->SetMinNClustersTPC(50);
}
esdTrackCuts->SetMaxChi2PerClusterTPC(4);
esdTrackCuts->SetAcceptKinkDaughters(kFALSE);
esdTrackCuts->SetRequireTPCRefit(kTRUE);
// ITS
esdTrackCuts->SetRequireITSRefit(kTRUE);
esdTrackCuts->SetClusterRequirementITS(AliESDtrackCuts::kSPD,
                                       AliESDtrackCuts::kAny);
if(selPrimaries) {
  // 7*(0.0015+0.0050/pt^1.1)
  esdTrackCuts->SetMaxDCAToVertexXYPtDep("0.0105+0.0350/pt^1.1");
  esdTrackCuts->SetMaxChi2TPCConstrainedGlobal(36);
}
esdTrackCuts->SetMaxDCAToVertexZ(2);
esdTrackCuts->SetDCAToVertex2D(kFALSE);
esdTrackCuts->SetRequireSigmaToVertex(kFALSE);

esdTrackCuts->SetMaxChi2PerClusterITS(36);
```



AliESDtrackCuts: TPC cuts

```
AliESDtrackCuts* esdTrackCuts = new AliESDtrackCuts;
// TPC
if(clusterCut == 0)  esdTrackCuts->SetMinNClustersTPC(50);
else if (clusterCut == 1) {
  esdTrackCuts->SetMinNCrossedRowsTPC(70);
  esdTrackCuts->SetMinRatioCrossedRowsOverFindableClustersTPC(0.8);
}
else {

  esdTrackCuts->SetMinNClustersTPC(50);
}
esdTrackCuts->SetMaxChi2PerClusterTPC(4);
esdTrackCuts->SetAcceptKinkDaughters(kFALSE);
esdTrackCuts->SetRequireTPCRefit(kTRUE);
// ITS
esdTrackCuts->SetRequireITSRefit(kTRUE);
esdTrackCuts->SetClusterRequirementITS(AliESDtrackCuts::kSPD,
                                       AliESDtrackCuts::kAny);
if(selPrimaries) {
  // 7*(0.0015+0.0050/pt^1.1)
  esdTrackCuts->SetMaxDCAToVertexXYPtDep("0.0105+0.0350/pt^1.1");
  esdTrackCuts->SetMaxChi2TPCConstrainedGlobal(36);
}
esdTrackCuts->SetMaxDCAToVertexZ(2);
esdTrackCuts->SetDCAToVertex2D(kFALSE);
esdTrackCuts->SetRequireSigmaToVertex(kFALSE);

esdTrackCuts->SetMaxChi2PerClusterITS(36);
```



$(y, z, \sin(\varphi), \tan(\lambda), 1/p_t)$

χ^2-difference between: TPConly track parameters extrapolated to the primary vertex and global track parameters. Removes fake high-pt tracks due to wrong association of ITS clusters.

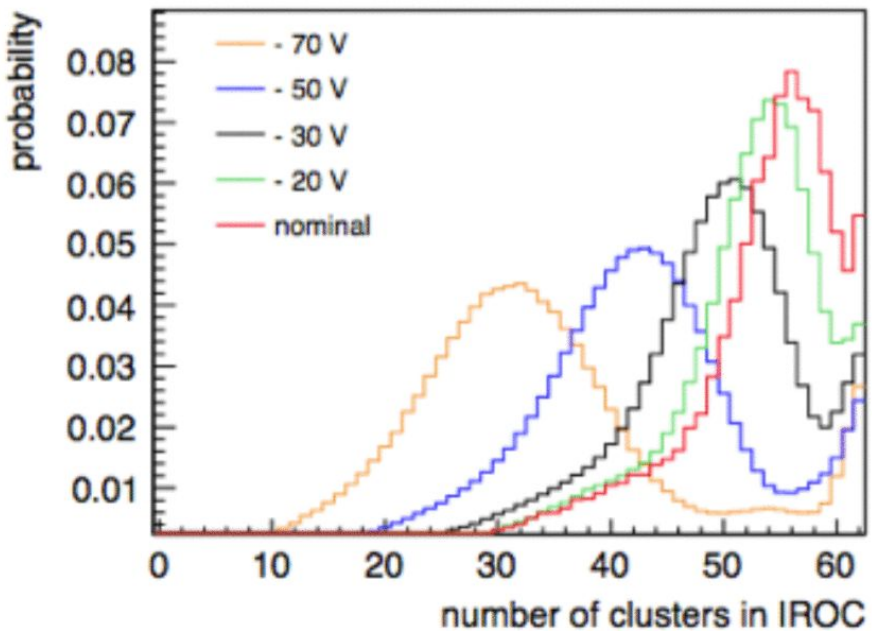
AliESDtrackCuts: ITS cuts

```
AliESDtrackCuts* esdTrackCuts = new AliESDtrackCuts;
// TPC
if(clusterCut == 0)  esdTrackCuts->SetMinNClustersTPC(50);
else if (clusterCut == 1) {
  esdTrackCuts->SetMinNCrossedRowsTPC(70);
  esdTrackCuts->SetMinRatioCrossedRowsOverFindableClustersTPC(0.8);
}
else {

  esdTrackCuts->SetMinNClustersTPC(50);
}
esdTrackCuts->SetMaxChi2PerClusterTPC(4);
esdTrackCuts->SetAcceptKinkDaughters(kFALSE);
esdTrackCuts->SetRequireTPCRefit(kTRUE);
// ITS
esdTrackCuts->SetRequireITSRefit(kTRUE);
esdTrackCuts->SetClusterRequirementITS(AliESDtrackCuts::kSPD,
                                       AliESDtrackCuts::kAny);
if(selPrimaries) {
  // 7*(0.0015+0.0050/pt^1.1)
  esdTrackCuts->SetMaxDCAToVertexXYPtDep("0.0105+0.0350/pt^1.1");
  esdTrackCuts->SetMaxChi2TPCConstrainedGlobal(36);
}
esdTrackCuts->SetMaxDCAToVertexZ(2);
esdTrackCuts->SetDCAToVertex2D(kFALSE);
esdTrackCuts->SetRequireSigmaToVertex(kFALSE);

esdTrackCuts->SetMaxChi2PerClusterITS(36);
```

Cut at ≈ 7σ of impact parameter resolution.
N.B.: resolution is only given for the tracks which fulfill the other cuts.

Track properties depend on detector conditions

- The effect of the cuts depends on the detector performance. Typical examples:
  - Phi-homogeneity of SPD-cluster cut depends on the number of active modules which changed with time.
  - Number of TPC clusters depends on the gain of the chambers which also changed with time.
  - Chamber boundaries: high momenta tracks close to the chamber boundaries have naturally less clusters which is difficult to reflect in MC. N.B.: these clusters are not used for PID (a cut on TPC-PID or *track->GetTPCsignalN()* partially removes tracks on the boundaries.



AODtrack selection: filter bits

- AOD tracks have a **filter-bit mask** (data member *fFilterMap*)

```
UInt_t        fFilterMap;        // filter information, one bit per set of cuts
```

  - Store information about whether the track satisfies standard sets of quality criteria
  - Each bit correspond to a given set of cuts, e.g.:

```
enum AODTrkFilterBits_t {
  kTrkTPCOnly            = BIT(0), // Standard TPC only tracks
  kTrkITSsa              = BIT(1), // ITS standalone
  kTrkITSConstrained     = BIT(2), // Pixel OR necessary for the electrons
  kTrkElectronsPID       = BIT(3),    // PID for the electrons
  kTrkGlobalNoDCA        = BIT(4), // standard cuts with very loose DCA
  kTrkGlobal             = BIT(5),  // standard cuts with tight DCA cut
  kTrkGlobalSDD          = BIT(6), // standard cuts with tight DCA but with requiring the first SDD
  kTrkTPCOnlyConstrained = BIT(7) // TPC only tracks: TPConly information constrained to SPD vertex
};
```

- N.B.: the filter bits can change from production to production. Documentation is available in this twiki: https://twiki.cern.ch/twiki/bin/viewauth/ALICE/PWGPPAODTrackCuts
- The usage of a filter bit is recommended but might not be enough for your analysis! **You should be aware what it corresponds to!**

AODtrack selection: filter bits

- Track selection can be done using the filter bits using these methods of AliAODTrack

```
Bool_t  TestFilterBit(UInt_t filterBit) const {
  return (Bool_t) ((filterBit & fFilterMap) != 0);}
Bool_t  TestFilterMask(UInt_t filterMask) const {
  return (Bool_t) ((filterMask & fFilterMap) == filterMask);}
```

- Examples:

```
for(Int_t iTrack=0;iTrack<nTracks;iTrack++){
 AliAODTrack *aodTrack = aod->GetTrack(iTrack);
 if(!aodTrack) continue;
 // filter bit 128 denotes TPC-only tracks, use only them
 if(!aodTrack->TestFilterBit(128)) continue;
```

```
Bool_t IsAcceptedTrack(const AliAODTrack *aodTrack) {
  if (!aodTrack) return kFALSE;
  if(!aodTrack->TestFilterMask(BIT(5))) return kFALSE; // standard TPCITS with tight DCA
```

```
if(!(aodtrack->TestFilterBit(AliAODTrack::kTrkGlobalNoDCA) ||
     aodtrack->TestFilterBit(AliAODTrack::kTrkITSsa))) return kFALSE;
```

Filter bits: where and how they are defined

The filter bit scheme used for a given production is defined by:

1) the AddTask macro $ALICE_ROOT/ANALYSIS/ESDfilter/macros/AddTaskESDFilter.C

```
Bool_t AddTrackCutsLHC10bcde(AliAnalysisTaskESDfilter* esdFilter);
Bool_t AddTrackCutsLHC10h(AliAnalysisTaskESDfilter* esdFilter);
Bool_t AddTrackCutsLHC11h(AliAnalysisTaskESDfilter* esdFilter);
Bool_t AddTrackCutsLHC15f(AliAnalysisTaskESDfilter* esdFilter);
Bool_t enableTPCOnlyAODTracksLocalFlag=kFALSE;

AliAnalysisTaskESDfilter *AddTaskESDFilter(Bool_t useKineFilter=kTRUE,
                                           Bool_t writeMuonAOD=kFALSE,
                                           Bool_t writeDimuonAOD=kFALSE, /*obsolete*/
                                           Bool_t usePhysicsSelection=kFALSE,
                                           Bool_t useCentralityTask=kFALSE, /*obsolete*/
                                           Bool_t enableTPCOnlyAODTracks=kFALSE,
                                           Bool_t disableCascades=kFALSE,
                                           Bool_t disableKinks=kFALSE,
                                           Int_t runFlag = 1500, // The first 2 digits are the year, the second
                                                                 //2 digits are used to distinguish sub-periods (if needed)
                                           Int_t  muonMCMode = 3  ,
                                           Bool_t useV0Filter=kTRUE,
                                           Bool_t muonWithSPDTracklets=kTRUE,
                                           Bool_t isMuonCaloPass=kFALSE,
                                           Bool_t addPCMv0s=kTRUE)
{
  // Creates a filter task and adds it to the analysis manager.
   // Get the pointer to the existing analysis manager via the static access method.
   //==============================================================================
   AliAnalysisManager *mgr = AliAnalysisManager::GetAnalysisManager();
   if (!mgr) {
```

4 main methods, defining different “schemes” used for different periods

Various parameters: “runFlag” defines which of the above methods is used

Filter bits: where and how they are defined

The filter bit scheme used for a given production is defined by:

1) the AddTask macro $ALICE_ROOT/ANALYSIS/ESDfilter/macros/AddTaskESDFilter.C
2) the parameters passed to the AddTask in AODtrain.C

Particularly relevant: run_flag

In the past: hard-coded in the version of the macro used for a given production → only way to retrieve it is to check the actual macro used (but cuts used can be checked also from AOD event, see later)

Now: the parameters are passed through the JDL and set in the ProcessEnvironment() method of the task

```
AliAnalysisTaskESDfilter *taskesdfilter =
            AddTaskESDFilter(useKFILTER,
                             iMUONcopyAOD,          // write Muon AOD
                             kFALSE,                // write dimuon AOD
                             kFALSE,                // usePhysicsSelection
                             kFALSE,                // centrality OBSOLETE
                             kTRUE,                 // enable TPS only tracks
                             kFALSE,                // disable cascades
                             kFALSE,                // disable kinks
                             run_flag,              // run flag (YY00)
                             3,                     // muonMCMode
                             //kTRUE,               // useV0Filter
                             kFALSE,                // useV0Filter - turned o
                             muonWithSPDTracklets,
                             isMuonCaloPass,
                             iPWGGAgammaconv);      // Add PCMV0
  AliEMCALGeometry::GetInstance("","");
```

Filter bits: where and how they are defined

DPG ALICE • Data Preparation Group

```
}

Bool_t AddTrackCutsLHC15f(AliAnalysisTaskESDfilter* esdfilter){
  //
  // filter cuts for RunII pp in 2015
  // basically a duplication of 11h, but with stricter cluster requirement
  //
  Printf("%s%d: Creating Track Cuts for LHC15f",(char*)__FILE__,__LINE__);
  //
  // Cuts on primary tracks
  AliESDtrackCuts* esdTrackCutsL = AliESDtrackCuts::GetStandardTPCOnlyTrackCuts();

  // ITS stand-alone tracks
  AliESDtrackCuts* esdTrackCutsITSsa = new AliESDtrackCuts("ITS stand-alone Track Cuts", "ESD Track Cuts");
  esdTrackCutsITSsa->SetRequireITSStandAlone(kTRUE);

  // Pixel OR necessary for the electrons
  AliESDtrackCuts *itsStrong = new AliESDtrackCuts("ITSorSPD", "pixel requirement for ITS");
  itsStrong->SetClusterRequirementITS(AliESDtrackCuts::kSPD, AliESDtrackCuts::kAny);

  // PID for the electrons
  AliESDpidCuts *electronID = new AliESDpidCuts("Electrons", "Electron PID cuts");
  electronID->SetTPCnSigmaCut(AliPID::kElectron, 3.5);

  // standard cuts with very loose DCA
  AliESDtrackCuts* esdTrackCutsH = AliESDtrackCuts::GetStandardITSTPCTrackCuts2011(kFALSE);
  esdTrackCutsH->SetMaxDCAToVertexXY(2.4);
  esdTrackCutsH->SetMaxDCAToVertexZ(3.2);
  esdTrackCutsH->SetDCAToVertex2D(kTRUE);

  // standard cuts with tight DCA cut
  AliESDtrackCuts* esdTrackCutsH2 = AliESDtrackCuts::GetStandardITSTPCTrackCuts2011();

  // standard cuts with tight DCA but with requiring the first SDD cluster instead of an SPD clus
  // tracks selected by this cut are exclusive to those selected by the previous cut
  AliESDtrackCuts* esdTrackCutsH3 = AliESDtrackCuts::GetStandardITSTPCTrackCuts2011();
  esdTrackCutsH3->SetClusterRequirementITS(AliESDtrackCuts::kSPD, AliESDtrackCuts::kNone);
  esdTrackCutsH3->SetClusterRequirementITS(AliESDtrackCuts::kSDD, AliESDtrackCuts::kFirst);

  // TPC only tracks: Optionally enable the writing of TPConly information
  // constrained to SPD vertex in the filter below
  AliESDtrackCuts* esdTrackCutsTPCOnly = AliESDtrackCuts::GetStandardTPCOnlyTrackCuts();
  // The following line is needed for 2010 PbPb reprocessing and pp, but not for 2011 PbPb
-:--  AddTaskESDFilter.C   81% L573   Git-master   (C++/l Abbrev)----------------------------------
```

(example, method AddTrackCutsLHC15f)

In these methods, a **series of AliESDtrackCuts** objects are created that are then used for a given filter bit

```
  // Compose the filter
  AliAnalysisFilter* trackFilter = new AliAnalysisFilter("trackFilter");
  // 1, 1<<0
  trackFilter->AddCuts(esdTrackCutsL);
  // 2, 1<<1
  trackFilter->AddCuts(esdTrackCutsITSsa);
  // 4, 1<<2
  trackFilter->AddCuts(itsStrong);
  itsStrong->SetFilterMask(1);        // AND with Standard track cuts
  // 8, 1<<3
  trackFilter->AddCuts(electronID);
  electronID->SetFilterMask(4);       // AND with Pixel Cuts
   // 16, 1<<4
  trackFilter->AddCuts(esdTrackCutsH);
  // 32, 1<<5
  trackFilter->AddCuts(esdTrackCutsH2);
  // 64, 1<<6
  trackFilter->AddCuts(esdTrackCutsH3);
  // 128 , 1 << 7
  trackFilter->AddCuts(esdTrackCutsTPCOnly);
  if(enableTPCOnlyAODTracksLocalFlag)esdfilter->SetTPCOnlyFilterMask(128);
  // 256, 1 << 8 Global Hybrids
  trackFilter->AddCuts(esdTrackCutsHTG);
  esdfilter->SetHybridFilterMaskGlobalConstrainedGlobal((1<<8)); // these normal global tracks will be marked as hybrid
  // 512, 1<< 9 GlobalConstraint Hybrids
  trackFilter->AddCuts(esdTrackCutsHTGC);
  esdfilter->SetGlobalConstrainedFilterMask(1<<9); // these tracks are written out as global constrained tracks
  esdfilter->SetWriteHybridGlobalConstrainedOnly(kTRUE); // write only the complement
  // 1024, 1<< 10 // tight DCA cuts
  trackFilter->AddCuts(esdTrackCutsH2Cluster);
  // 2048, 1<<11 // duplication of 1<<5 with looser CrossedRows requirements for forward eta
  trackFilter->AddCuts(esdTrackCutsH2Forward);
  // 4096, 1<<12 // duplication of 1<<6 with looser CrossedRows requirements for forward eta
  trackFilter->AddCuts(esdTrackCutsH3Forward);

  esdfilter->SetTrackFilter(trackFilter);

  return kTRUE;

}
```

These cut objects are then passed to the an **AliAnalysisFilter** object, which can be considered a sort-of manager class for a list of cuts, and this object is set to the AliAnalysisTaskESDfilter object created in the AddTask

Filter bits: main methods

Track cuts used for filter bits are defined mainly by the following methods (look e.g. at the AddTrackCutsLHC15f method shown before)**:**

1) AliESDtrackCuts::GetStandardTPCOnlyTrackCuts(): loose "baseline" TPC-only selection, it provides a quite uniform φ distribution but it keeps also lots of secondary tracks (from decays and interaction with material), pile-up tracks, badly reconstructed tracks. It is used for filter bit 0 (n.b. 2^0=1).

```
AliESDtrackCuts* esdTrackCuts = new AliESDtrackCuts;
  esdTrackCuts->SetMinNClustersTPC(50);
  esdTrackCuts->SetMaxChi2PerClusterTPC(4);
  esdTrackCuts->SetAcceptKinkDaughters(kFALSE);
  esdTrackCuts->SetMaxDCAToVertexZ(3.2);
  esdTrackCuts->SetMaxDCAToVertexXY(2.4);
  esdTrackCuts->SetDCAToVertex2D(kTRUE);
```

Filter bits: main methods

2) AliESDtrackCuts::GetStandardITSTPCTrackCuts2011(Bool_t selPrimaries, Int_t clusterCut); (default method parameters are kTRUE,1).

TPC+ITS(SPD) tracks: baseline for most filter bits (e.g. 4,5)

```
esdTrackCuts->SetMaxChi2PerClusterTPC(4);
esdTrackCuts->SetAcceptKinkDaughters(kFALSE);
esdTrackCuts->SetRequireTPCRefit(kTRUE);
esdTrackCuts->SetRequireITSRefit(kTRUE);
esdTrackCuts->SetClusterRequirementITS(AliESDtrackCuts::kSPD, AliESDtrackCuts::kAny);
esdTrackCuts->SetMaxChi2PerClusterITS(36);
esdTrackCuts->SetMaxDCAToVertexZ(2);
esdTrackCuts->SetDCAToVertex2D(kFALSE);
esdTrackCuts->SetRequireSigmaToVertex(kFALSE);
```

ITS is required, SPD::kAny

Can introduce modulations in η,φ induced by dead modules in SPD

```
if(clusterCut == 0)  esdTrackCuts->SetMinNClustersTPC(50);
 else if (clusterCut == 1) {
  esdTrackCuts->SetMinNCrossedRowsTPC(70);
  esdTrackCuts->SetMinRatioCrossedRowsOverFindableClustersTPC(0.8);
 }
```

Switch for selection on TPC clusters or (crossed rows & crossed rows/ findable clusters)

```
if(selPrimaries) {
        // 7*(0.0015+0.0050/pt^1.1)
        esdTrackCuts->SetMaxDCAToVertexXYPtDep("0.0105+0.0350/pt^1.1");
        esdTrackCuts->SetMaxChi2TPCConstrainedGlobal(36);
 }
```

Switch for applying a tight cut on DCA (~7σ) for selecting primary particles + "golden" χ^2 cut

Filter bits: list and main features

Remark: the main features and goals of the bits are quite stable vs. time. The practical implementation and specific selections may instead vary with time! What reported here corresponds to what used for pp 2016 ("AddTrackCutsLHC15f" method of AddTaskESDFilter.C). In red: bits more widely used.

Filter-bit 0

Baseline, very loose TPC-only selection. φ distribution should be rather flat, unless of issues in the TPC.

```
AliESDtrackCuts* esdTrackCutsL = AliESDtrackCuts::GetStandardTPCOnlyTrackCuts();
```

Filter-bit 1: ITS stand-alone tracks

```
AliESDtrackCuts* esdTrackCutsITSsa = new AliESDtrackCuts("ITS stand-alone Track Cuts", "ESD Track Cuts");
esdTrackCutsITSsa->SetRequireITSStandAlone(kTRUE);
```

Filter-bit 2: TPC only (filter bit 0) + SPDor

```
AliESDtrackCuts *itsStrong = new AliESDtrackCuts("ITSorSPD", "pixel requirement for ITS");
itsStrong->SetClusterRequirementITS(AliESDtrackCuts::kSPD, AliESDtrackCuts::kAny);
itsStrong->SetFilterMask(1);     // AND with Standard track cuts
```

Adds selection from previous filter bit, in this case is from bit 0

This bit is TPConly + SPDany → it has inhomogeneities in φ

Filter bits: list and main features

Filter-bit 3: as filter-bit 2 + PID for the electrons

```
AliESDpidCuts *electronID = new AliESDpidCuts("Electrons", "Electron PID cuts");
electronID->SetTPCnSigmaCut(AliPID::kElectron, 3.5);
electronID->SetFilterMask(4);          // AND with Pixel Cuts
```

Filter-bit 4

Standard TPC+ITS(SPD) cuts with very loose DCA. It introduces inhomogeneities in φ.

```
AliESDtrackCuts* esdTrackCutsH=AliESDtrackCuts::GetStandardITSTPCTrackCuts2011(kFALSE);
```

Plus/overwriting:

```
esdTrackCutsH->SetMaxDCAToVertexXY(2.4);
esdTrackCutsH->SetMaxDCAToVertexZ(3.2);
esdTrackCutsH->SetDCAToVertex2D(kTRUE);
```

Filter-bit 5

Standard TPC+ITS cuts with tight DCA cut → **can be used for selecting primary particles (but can reject tracks from charm/beauty decay).** It introduces inhomogeneities in φ.

```
AliESDtrackCuts* esdTrackCutsH2=AliESDtrackCuts::GetStandardITSTPCTrackCuts2011(kTRUE);
```

Tight DCA cut and golden χ^2 cut!

Filter bits: list and main features

Filter-bit 6 (= filter-bit 5 but requiring a point in SDD and no points in SPD → no intersection with filter-bit 5)

It should provide highly inhomogeneous track φ,η distributions but, combined with filter-bit 5 should provide a quite uniform φ,η coverage. However, it has to be noted that, since there are no SPD points, the track dca distribution is worse than for filter-bit 5 and the interplay of this selection with the golden χ^2 cut could be tricky. It could cause a lower efficiency for primary particles and a less effective rejection of secondary particles w.r.t filter-bit 5.

```
AliESDtrackCuts* esdTrackCutsH3=AliESDtrackCuts::GetStandardITSTPCTrackCuts2011(kTRUE);
 esdTrackCutsH3->SetClusterRequirementITS(AliESDtrackCuts::kSPD, AliESDtrackCuts::kNone);
 esdTrackCutsH3->SetClusterRequirementITS(AliESDtrackCuts::kSDD, AliESDtrackCuts::kFirst);
```

Filter-bit 7: tpc-only tracks.

```
// TPC only tracks: Optionally enable the writing of TPConly information
 // constrained to SPD vertex in the filter below
AliESDtrackCuts* esdTrackCutsTPCOnly = AliESDtrackCuts::GetStandardTPCOnlyTrackCuts();
if(enableTPCOnlyAODTracksLocalFlag)esdfilter->SetTPCOnlyFilterMask(128);
```

“Hybrid tracks”=filter bit 8 +9

Goal: select mostly primary tracks but guarantee a uniform φ,η distribution

Filter bit 8

```
AliESDtrackCuts* esdTrackCutsHTG=AliESDtrackCuts::GetStandardITSTPCTrackCuts2011(kFALSE);
 esdTrackCutsHTG->SetName("Global Hybrid tracks, loose DCA");
 esdTrackCutsHTG->SetMaxDCAToVertexXY(2.4);
 esdTrackCutsHTG->SetMaxDCAToVertexZ(3.2);
 esdTrackCutsHTG->SetDCAToVertex2D(kTRUE);
 esdTrackCutsHTG->SetMaxChi2TPCConstrainedGlobal(36);
 esdTrackCutsHTG->SetMaxFractionSharedTPCClusters(0.4);
```

loose DCA cut!

But golden χ^2 cut

Filter bit 9

```
// Complementary tracks which will be stored as global constraint, complement is done in the ESDFilter task
 AliESDtrackCuts* esdTrackCutsHTGC = new AliESDtrackCuts(*esdTrackCutsHTG);
 esdTrackCutsHTGC->SetName("Global Constraint Hybrid tracks, loose DCA no it requirement");
 esdTrackCutsHTGC->SetClusterRequirementITS(AliESDtrackCuts::kSPD,AliESDtrackCuts::kOff);
 esdTrackCutsHTGC->SetRequireITSRefit(kTRUE);
```

```
// 256, 1 << 8 Global Hybrids
 trackFilter->AddCuts(esdTrackCutsHTG);
 esdfilter->SetHybridFilterMaskGlobalConstrainedGlobal((1<<8)); // these normal global tracks will be marked as
hybrid
 // 512, 1<< 9 GlobalConstraint Hybrids
 trackFilter->AddCuts(esdTrackCutsHTGC);
 esdfilter->SetGlobalConstrainedFilterMask(1<<9); // these tracks are written out as global constrained tracks
 esdfilter->SetWriteHybridGlobalConstrainedOnly(kTRUE); // write only the complement
```

Filter bits: list and main features

Filter-bit 10: as filter-bit 5 but using cluster cut instead of crossed rows (à la 2010 default)

```
AliESDtrackCuts* esdTrackCutsH2Cluster = AliESDtrackCuts::GetStandardITSTPCTrackCuts2011(kTRUE, 0);
esdTrackCutsH2Cluster->SetMinNClustersTPC(70); // gain in 2015 is higher than in 2011
```

Filter-bit 11, 12: duplication of filter-bit 5 and 6 but with looser requirement on CrossedRows and CrossedRowsOverFindable in order to go to forward eta (To be used with care!)

Filter-bit 11:

```
AliESDtrackCuts* esdTrackCutsH2Forward = AliESDtrackCuts::GetStandardITSTPCTrackCuts2011();
esdTrackCutsH2Forward->SetMinNCrossedRowsTPC(50);
esdTrackCutsH2Forward->SetMinRatioCrossedRowsOverFindableClustersTPC(0.6);
```

Filter-bit 12:

```
AliESDtrackCuts* esdTrackCutsH3Forward = AliESDtrackCuts::GetStandardITSTPCTrackCuts2011();
esdTrackCutsH3Forward->SetMinNCrossedRowsTPC(50);
esdTrackCutsH3Forward->SetMinRatioCrossedRowsOverFindableClustersTPC(0.6);
esdTrackCutsH3Forward->SetClusterRequirementITS(AliESDtrackCuts::kSPD, AliESDtrackCuts::kNone);
esdTrackCutsH3Forward->SetClusterRequirementITS(AliESDtrackCuts::kSDD, AliESDtrackCuts::kFirst);
```

How to retrieve filter cuts from aod

Inside the AliAnalysisTaskESDfilter::UserCreateOutputObjects() the AliAnalysisFilter object to which the cut objects used to define the filter bits are added is saved in the "UserInfo" TList attached to the aodTree.

It is very easy to retrieve the cut objects used for a given filter bit

```
void AliAnalysisTaskESDfilter::UserCreateOutputObjects()
{
  //
  // Create Output Objects conenct filter to outputtree
  //
  if(OutputTree())
  {
    OutputTree()->GetUserInfo()->Add(fTrackFilter);
  }
  else
  {
    AliError("No OutputTree() for adding the track filter");
  }
```

It can be done both opening an aod file or in a task during analysis. The only difference is how to get the pointer to the **aod tree**:

From a file

```
TFile *f=TFile::Open("run125632numb037/AliAOD.root")
TTree *aodt=(TTree*)f->Get("aodTree")
```

During analysis, from the aod handler

```
AliAODHandler *aodHandler =
  dynamic_cast<AliAODHandler*>(AliAnalysisManager::GetAnalysisManager()->GetOutputEventHandler());
if (aodHandler) {
  TTree *aodTree = aodHandler->GetTree();
}
```

Then, same few steps:

```
TList *lt=aodTree->GetUserInfo();
```

Get UserInfo TList

```
AliAnalysisFilter *flt=(AliAnalysisFilter *)lt->FindObject("trackFilter")
TList *lisCuts=flt->GetCuts()
```

Get Analysis Filter object
Get TList with cuts used

How to retrieve filter cuts from aod

```
[root [6] lisCuts->Print()
 Collection name='TList', class='TList', size=11
  OBJ: AliESDtrackCuts   AliESDtrackCuts                                          FB 0
  OBJ: AliESDtrackCuts   ITS stand-alone Track Cuts      ESD Track Cuts           FB 1
  OBJ: AliESDtrackCuts   ITSorSPD        pixel requirement for ITS                .
  OBJ: AliESDpidCuts     Electrons       Electron PID cuts                        .
  OBJ: AliESDtrackCuts   AliESDtrackCuts                                          .
  OBJ: AliESDtrackCuts   AliESDtrackCuts                                          .
  OBJ: AliESDtrackCuts   AliESDtrackCuts                                          .
  OBJ: AliESDtrackCuts   AliESDtrackCuts                                          .
  OBJ: AliESDtrackCuts   Global Hybrid tracks, loose DCA                          .
  OBJ: AliESDtrackCuts   Global Constraint Hybrid tracks, loose DCA no it requirement  FB 9
  OBJ: AliESDtrackCuts   AliESDtrackCuts                                          FB 10
 root [7]
```

Order of the cuts correspond to filter bit number → easy to retrieve the esd-track-cut object for a given filter bit

```
AliESDtrackCuts *trk=(AliESDtrackCuts*)ltc->At(filterbitnumber);
```

To check the cut values one can **Dump** the object or use the **dedicated getters**

```
root [8] trk->Dump()
==> Dumping object at: 0x00007fc0e8525110, name=AliESDtrackCuts, class=AliESDtrackCuts

fCutMinNClusterTPC            50                min number of tpc clusters
fCutMinNClusterITS            -1                min number of its clusters
fCutMinNCrossedRowsTPC        -1                min number of tpc crossed rows
fCutMinRatioCrossedRowsOverFin-1bleClustersTPC  min ratio crossed rows / findable clusters
*f1CutMinNClustersTPCPtDep    ->0               pt dependent tpc clusters cut
fCutMaxPtDepNClustersTPC      0                 maximum pt for pt dependend TPC cluster cut. For pt=>ptmax NClusterMin = f1CutMinNClustersTPCPtDep->Eval(fCutMaxPtDepNClustersTPC).
fCutMinLengthActiveVolumeTPC  0                 mininum length (in cm) over which the track is sampled in the active volume of the TPC (outside boundaries)
fDeadZoneWidth                0                 width of the TPC dead zone (missing pads + PRF +ExB)
fCutGeoNcrNclLength           0                 cut on the geometical length  condition Ngeom(cm)>cutGeoNcrNclLength default=130
fCutGeoNcrNclGeom1Pt          0                 1/pt dependence slope  cutGeoNcrNclLength:=fCutGeoNcrNclLength-abs(1/pt)^fCutGeoNcrNclGeom1Pt
fCutGeoNcrNclFractionNcr      0                 relative fraction cut Ncr  condition Ncr>cutGeoNcrNclFractionNcr*fCutGeoNcrNclLength
fCutGeoNcrNclFractionNcl      0                 ralative fraction cut Ncr  condition Ncl>cutGeoNcrNclFractionNcl
fCutOutDistortedRegionTPC     false             flag if distorted regions in the TPC should be cut out
fCutClusterRequirementITS[3]  0                 detailed ITS cluster requirements for (SPD, SDD, SSD)
fCutMaxChi2PerClusterTPC      4                 max tpc fit chi2 per tpc cluster
fCutMaxChi2PerClusterITS      1e+10             max its fit chi2 per its cluster
fCutMaxChi2TPCConstrainedVsGlo1e+10             max chi2 TPC track constrained with vtx vs. global track
fCutMaxChi2TPCConstrainedVsGlo3alVertexType     vertex type for max chi2 TPC track constrained with vtx vs. global track (can be configured to accept several vertex types)
fCutMaxMissingITSPoints       6                 max n. of missing ITS points
fCutMaxC11                    1e+10             max cov. matrix diag. elements (res. y^2)
```

Methods to make this more immediate and allow for a fast comparison between different esd track cut objects are being developed

TRACKING: PERFORMANCE and EFFICIENCY

Standard procedure for efficiency systematic uncertainties

https://twiki.cern.ch/twiki/bin/view/ALICE/AliDPGtoolsTrackSystematicUncertainty

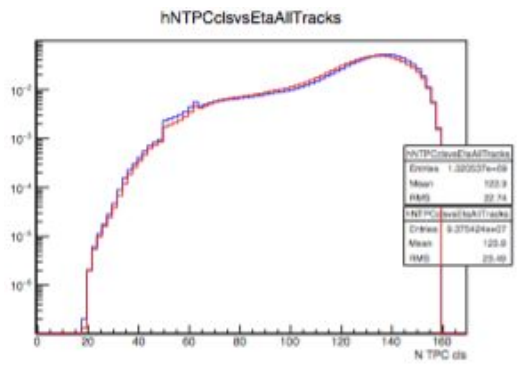
- Systematic on tracking efficiency estimated from two contributions:
  1. Track finding and selection in the TPC
     $\longrightarrow$ estimated by TPC cut variation
  2. TPC-ITS matching (prolongation with a specific set of selections in ITS)
     $\longrightarrow$ estimated by comparing matching efficiency in data and MC

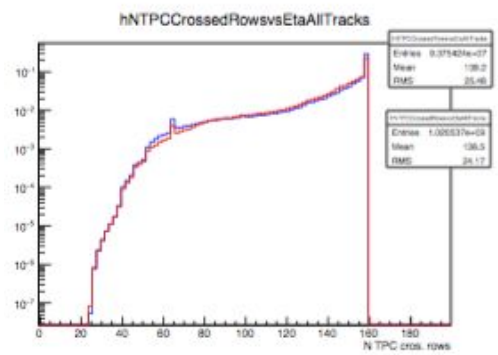
Variation of TPC track selections

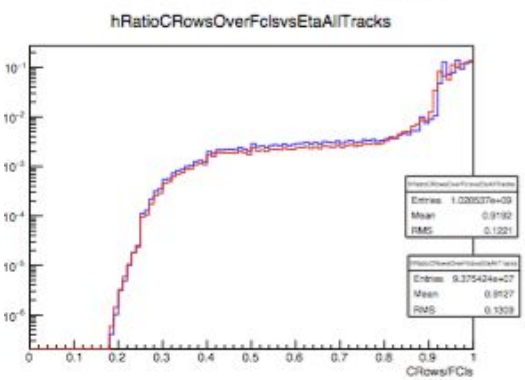
- This is an example, we will try to propose a standard procedure, but may depend on analysis
- Stdcuts: n_{TPCcls}>70; χ^2/cl<4; CrRows/Findb>0.8; |η|<0.8
- Variations: add/remove tighter cuts
  - stdcuts + TPC crossed rows > 120 - (5/pT)
  - stdcuts + TPC N clusters > 0.5 * TPC crossed rows
  - stdcuts + TPC signal N > 0.5 * TPC crossed rows
  - stdcuts + TPC crossed rows/TPC findable clusters > 0.9
- Compare corrected yields

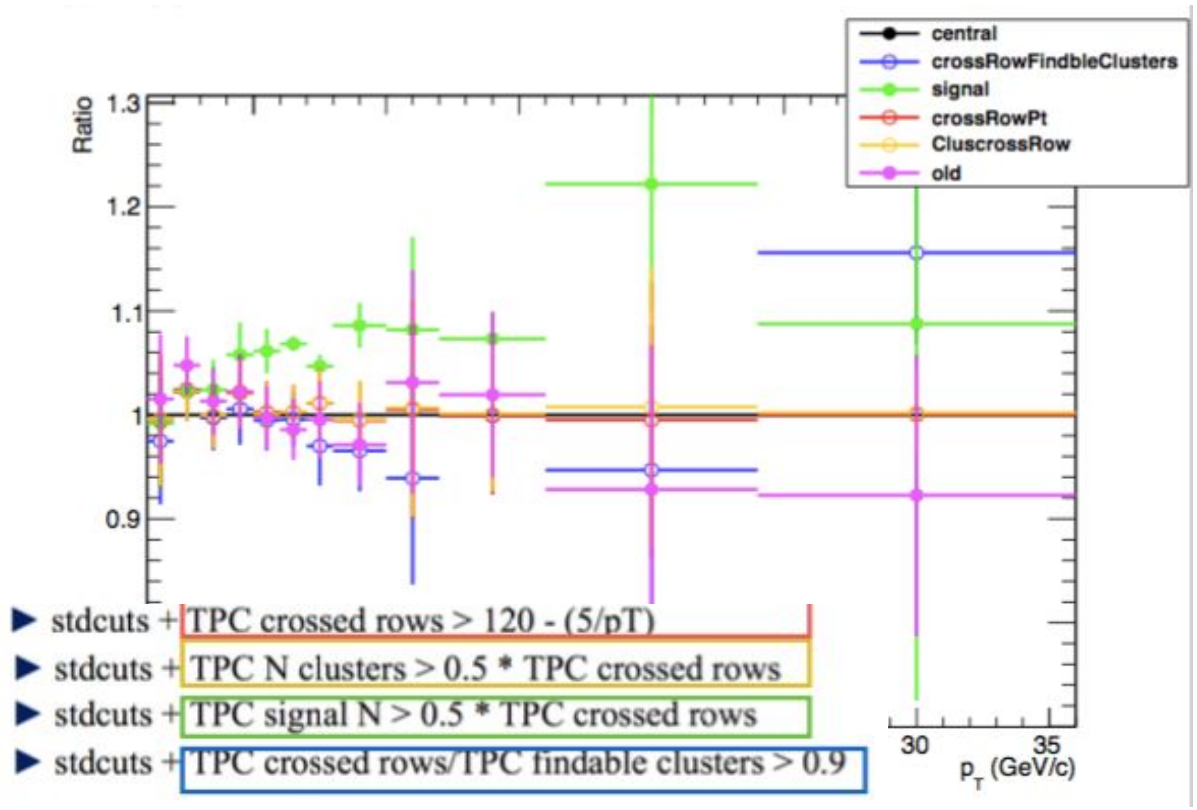
Variation of TPC track selections: example for D^0 cross section in pp 7 TeV

(results from pass 4 data)





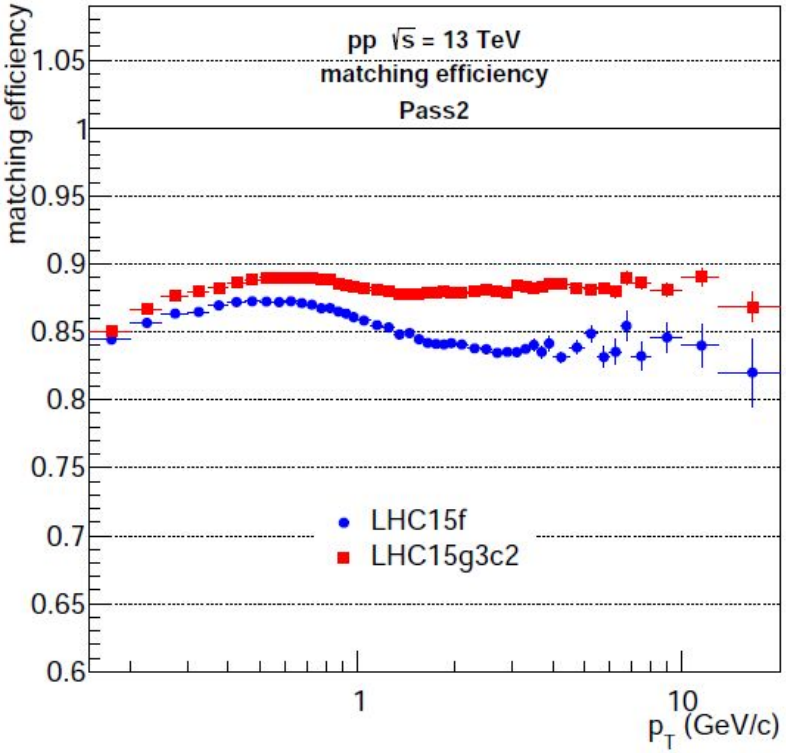




We could provide these comparison plots for selected datasets in the DPG twiki

Matching efficiency systematic unc.

- A contribution to the systematic uncertainty on the tracking efficiency is commonly estimated by comparing the fraction of TPC tracks prolonged to ITS (+ hit in SPD) in data and MC (matching efficiency)
- The matching efficiency for charged tracks can differ significantly between data and MC (up to 6-8%)
- However: the comparison is affected not only by how well the MC describes the tracking, but also by the relative contributions of primary and secondary particles in data and MC



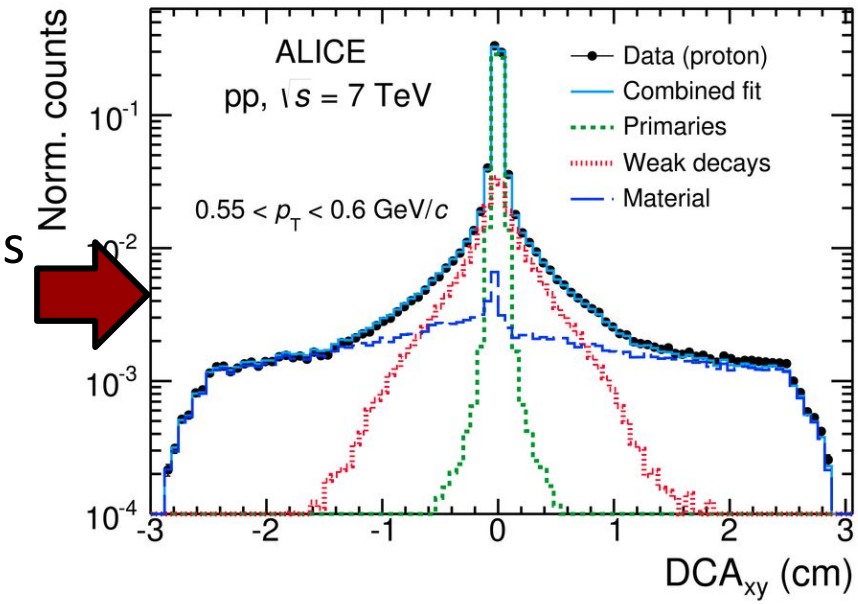
Physical primary particles

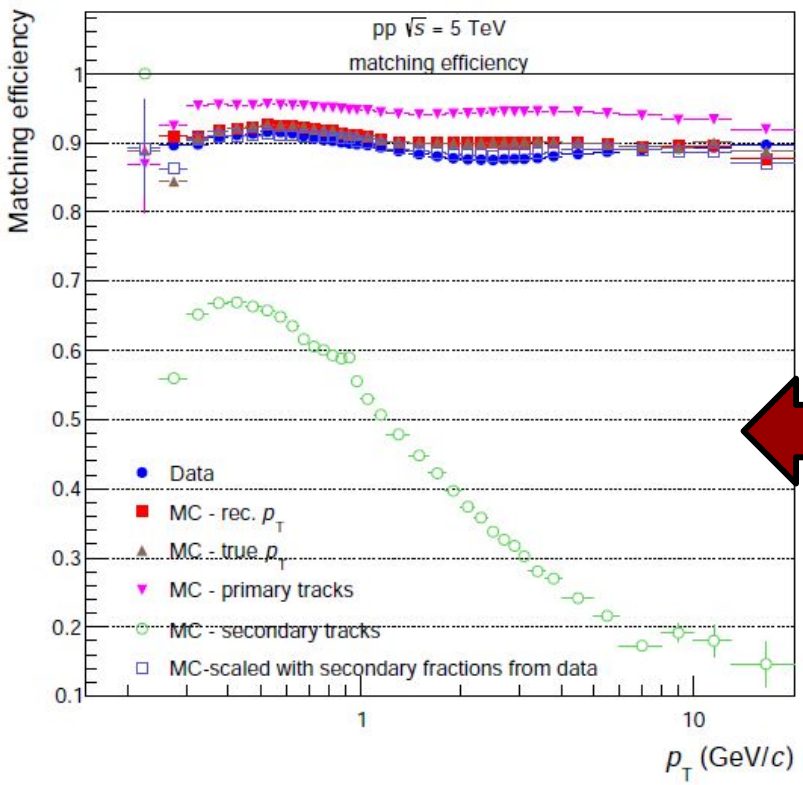
- **Physical primary particles:**
  - Definition: *A primary particle is a particle with a mean proper lifetime τ larger than 1 cm/c, which is either a) produced directly in the interaction, or b) from decays of particles with τ smaller than 1 cm/c*
  - Tagging of primaries implemented in AliRoot code in:
    - ESD/Kinematics: AliStack::IsPhysicalPrimary()
    - AOD: AliAODMCParticle::IsPhysicalPrimary()

- **Secondary particles:** distinguished in two categories
  - Produced in **weak decays of long-lived (strange) hadrons**
    - Tagged with: AliStack::IsSecondaryFromWeakDecay
  - Produced in **interactions in the detector material**
    - Tagged with: AliStack::IsSecondaryFromMaterial

Primary vs. secondary particles

- **Impact parameter** (Distance of Closest approach of track to interaction vertex) distribution
  - Peaked at 0 for primary particles
  - Wider for secondaries





- TPC->ITS track prolongation efficiency (**matching efficiency**)
  - Substantially lower matching efficiency for secondaries than for primaries

Matching efficiency systematic unc.

- Defined from the ratio of matching efficiencies in data and MC, with fraction of secondaries "equalised" in data and MC using DCA fits
  - Procedure introduced for charged-particle p_T spectra and R_{AA} analysis (https://aliceinfo.cern.ch/Notes/node/472)
  - Provided a task for general use (selections can be changed): PWGPP/EvTrkSelection/AliAnalysisTrackingUncertaintiesAOT

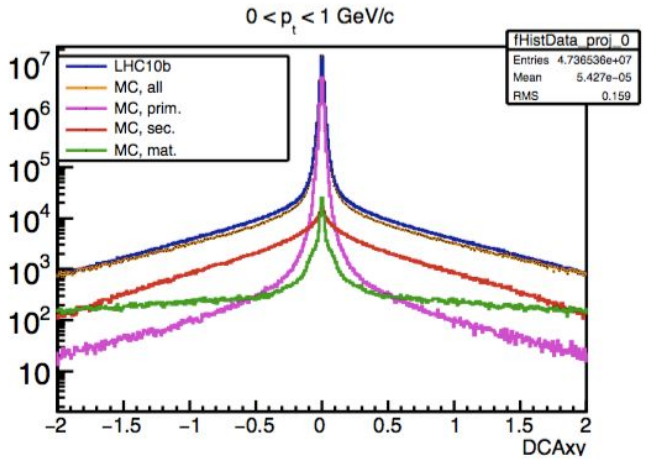
3 main ingredients:

1) **Matching efficiencies for particle types:** $Eff^{MC}_{primaries}$, $Eff^{MC}_{secondaries}$, $Eff^{Data}_{inclusive}$
2) **Primary fraction from data:** $f'_{primaries}$
3) **Combine into inclusive efficiency:**

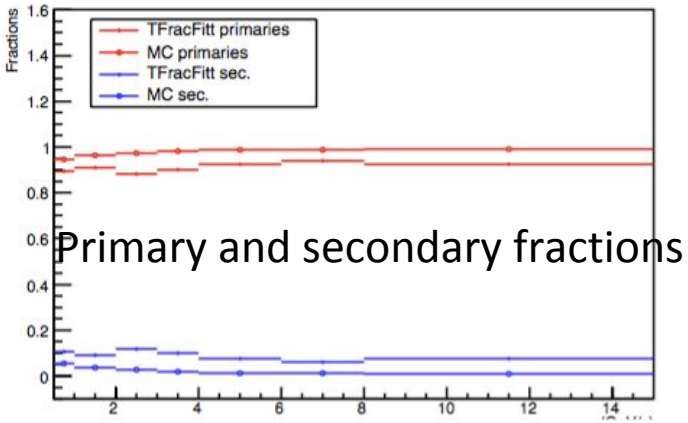
$$Eff^{MC}_{inclusive} = f'_{primaries} \times Eff^{MC}_{primaries} + (1 - f'_{primaries}) \times Eff^{MC}_{secondaries}$$

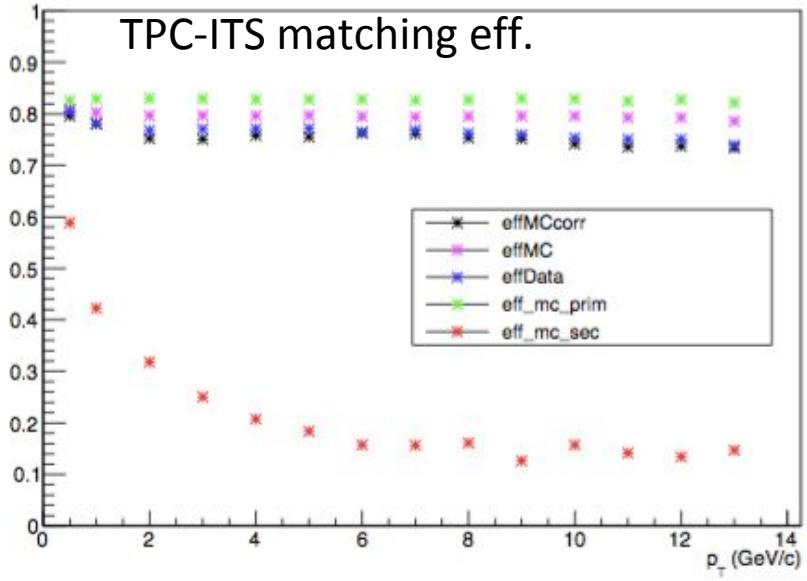
Systematic uncertainty: $(Eff^{Data}_{inclusive} - Eff^{MC}_{inclusive})/Eff^{Data}_{inclusive}$

Example: pp 7 TeV (pass 4)

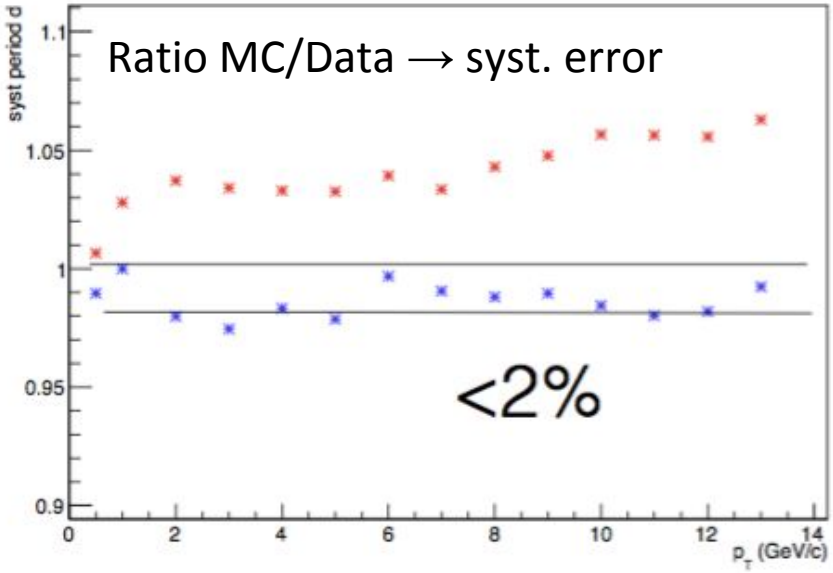


→





→



IF YOU NEED MORE INFORMATION

DPG twiki pages

- More detailed information can be found in: https://twiki.cern.ch/twiki/bin/viewauth/ALICE/AliceDPG

- Any feedback on these twiki pages is more than welcome
  - It will help us to improve them and keeping them up-to-date
- For any question, doubt, suggestion, constructive criticism, contact us: alice-dpg-coordination@cern.ch