

Using skimmed/reduced trees and nano-AODs in DQ analysis

I.Arsene on behalf of DQ
University of Oslo
2017/11/08

Input from: R.Arnaldi, I.Das, P.Dillenseger, T.Gunji, T.Jimenez-Bustamante, S.Lehner, P.Pilot,
A.Uras, O.Vazquez-Doce, M.Weber, S.Weber, D.Weisser

Motivation

- Typical DQ analysis involve measurements of quarkonia decaying in dilepton decay channels or low mass dileptons
 - Leptons represent a very small fraction of the tracks written in ESD/AODs in the central barrel
 - For quarkonia analyses, we only need tracks with relatively high p_t ($p_t > 1\text{GeV}/c$ for J/ψ)
 - Detector information needed in a specific analysis is usually much smaller than what is written in ESDs/AODs
- Producing data sets (“standard” or not) which contain JUST the needed information lead to (sometimes) huge reduction factors in data size and very fast processing

MUON analyses

- MUON analyses → special case because the datasets (ESD/AOD) are already relatively small
 - Most analyses use AliAOD.Muons.root
- There are a few examples of using skimmed trees
 - Low mass di-muons ([Antonio Uras](#))
 - J/psi to di-muons ([Roberta Arnaldi](#))
- Trees are produced in an analysis task on the grid
- Use LEGO trains or user analysis on AliAOD.Muons.root
- Reduction factor wrt AliAOD.Muons.root of ~2 (low mass dimuons) or ~20 (jpsi analyses)
 - The difference comes from the selection cuts (typically much more open for low mass dimuons)

Example of tree structure (Roberta)

- Small skimmed tree
- Contents: **event**, **track** and **di-muon** information
- Selection: standard MUON cuts. No fine tuning needed at the time of tree making
- **Example: run 265694 (p-Pb at 8.16 TeV)**
 - **AliAOD.root: 60 GB**
 - **AliAOD.Muons.root: 14 GB**
 - **Private tree: ~0.7 GB**
- Full p-Pb and Pb-p analysis on a private laptop takes ~2h

```
fOutputTree = new TTree("pPbTree", "Data Tree");
fOutputTree->Branch("FiredTriggerClasses", fTrigClass, "FiredTriggerClasses/C");
fOutputTree->Branch("inpmask", &finpmask, "inpmask/I"); //unsigned integer
fOutputTree->Branch("NMuons", &fNMuons, "NMuons/I");
fOutputTree->Branch("NContributors", &fNContributors, "NContributors/I");
fOutputTree->Branch("NTracklets", &fNTracklets, "NTracklets/I");
fOutputTree->Branch("Vertex", fVertex, "Vertex[3]/D");
fOutputTree->Branch("PercentV0M", &fPercentV0M, "PercentV0M/F");
fOutputTree->Branch("PercentCL0", &fPercentCL0, "PercentCL0/F");
fOutputTree->Branch("PercentCL1", &fPercentCL1, "PercentCL1/F");
fOutputTree->Branch("PercentV0A", &fPercentV0A, "PercentV0A/F");
fOutputTree->Branch("PercentV0C", &fPercentV0C, "PercentV0C/F");
fOutputTree->Branch("PercentZNA", &fPercentZNA, "PercentZNA/F");
fOutputTree->Branch("PercentZNC", &fPercentZNC, "PercentZNC/F");
fOutputTree->Branch("Pt", fPt, "Pt[NMuons]/D");
fOutputTree->Branch("E", fE, "E[NMuons]/D");
fOutputTree->Branch("Px", fPx, "Px[NMuons]/D");
fOutputTree->Branch("Py", fPy, "Py[NMuons]/D");
fOutputTree->Branch("Pz", fPz, "Pz[NMuons]/D");
fOutputTree->Branch("Y", fY, "Y[NMuons]/D");
fOutputTree->Branch("Eta", fEta, "Eta[NMuons]/D");
fOutputTree->Branch("MatchTrig", fMatchTrig, "MatchTrig[NMuons]/I");
fOutputTree->Branch("TrackChi2", fTrackChi2, "TrackChi2[NMuons]/D");
fOutputTree->Branch("MatchTrigChi2", fMatchTrigChi2, "MatchTrigChi2[NMuons]/D");
fOutputTree->Branch("DCA", fDCA, "DCA[NMuons]/D");
fOutputTree->Branch("Charge", fCharge, "Charge[NMuons]/I");
fOutputTree->Branch("RAtAbsEnd", fRAtAbsEnd, "RAtAbsEnd[NMuons]/D");
fOutputTree->Branch("NDimu", &fNDimu, "NDimu/I");
fOutputTree->Branch("DimuMu", fDimuMu, "DimuMu[NDimu][2]/I");
fOutputTree->Branch("DimuPt", fDimuPt, "DimuPt[NDimu]/D");
fOutputTree->Branch("DimuPx", fDimuPx, "DimuPx[NDimu]/D");
fOutputTree->Branch("DimuPy", fDimuPy, "DimuPy[NDimu]/D");
fOutputTree->Branch("DimuPz", fDimuPz, "DimuPz[NDimu]/D");
fOutputTree->Branch("DimuY", fDimuY, "DimuY[NDimu]/D");
fOutputTree->Branch("DimuMass", fDimuMass, "DimuMass[NDimu]/D");
fOutputTree->Branch("DimuCharge", fDimuCharge, "DimuCharge[NDimu]/I");
fOutputTree->Branch("DimuMatch", fDimuMatch, "DimuMatch[NDimu]/I");
```

Central barrel

- Both low mass dielectron and quarkonia analyses benefit from large rejection factors by filtering electrons
- A few approaches
 - Filtered AODs (aka nano-AODs)
 - Skimmed/reduced trees (a few implementations)
- Produced from either ESD, AOD or nano AODs using LEGO trains on the grid
- Need to be created 2-3 times during an analysis
- Typical data size reduction factors: 500-1000 relative to AOD size

Nano-AODs (used in both LMee and Jpsi2ee)

- Filtering task:
 - AliPhysics/PWGDQ/dielectron/core/AliAnalysisTaskDielectronFilter.cxx
- AddTask example ([Pascal Dillenseger](#))
 - AliPhysics/PWGDQ/dielectron/macros/AddTask_pdill_JpsiFilter_PbPb.C
- Standard event selection (PhysicsSelection, $-10 < z < +10$ cm, etc)
- Track selection: electrons
- Not needed branches removed

Skimmed trees (S.Lehner)

- Trees tailored for specific analyses
- Tree structure:
 - PWGDQ/Lmee/AliAnalysisTaskMLTreeMaker
 - PWGDQ/dielectron/macrosLMEE/AddTaskMLTreeMaker.
- Produced from nanoAODs, 2-3 times a year
- Contents: event and track selected information
 - Standard event selection
 - Track selection: electrons
- Example:
 - LHC15o_highIR, 40 million events
 - Size: 74 GB

```
fTree->Branch("centrality", &cent);
fTree->Branch("#tracks", &n);
fTree->Branch("pt", &pt);
fTree->Branch("eta", &eta);
fTree->Branch("phi", &phi);
fTree->Branch("charge", &charge);
fTree->Branch("RunNumber", &runn);
fTree->Branch("EsigTPC", &EsigTPC);
fTree->Branch("EsigITS", &EsigITS);
fTree->Branch("EsigTOF", &EsigTOF);
fTree->Branch("PsigTPC", &PsigTPC);
// fTree->Branch("NClustersITS", &NClustersITS);
fTree->Branch("NCrossedRowsTPC", &NCrossedRowsTPC);
fTree->Branch("NClustersTPC", &NClustersTPC);
fTree->Branch("RatioCrossedRowsFindableClusters", &RatioCrossedRowsFindableClusters);
fTree->Branch("HasSPDfirstHit", &HasSPDfirstHit);
fTree->Branch("NTPCSignal", &NTPCSignal);
```

```
fTree->Branch("DCAxy", &dcar);
fTree->Branch("DCAz", &dcaz);
fTree->Branch("vertx", &vertx);
fTree->Branch("verty", &verty);
fTree->Branch("vertz", &vertz);
fTree->Branch("nITS", &nITS);
fTree->Branch("nITSshared_frac", &nITSshared);
fTree->Branch("chi2ITS", &chi2ITS);
// fTree->Branch("chi2TPC", &chi2TPC);
fTree->Branch("chi2GlobalvsTPC", &chi2GlobalvsTPC);
fTree->Branch("chi2GlobalPerNDF", &chi2GlobalPerNDF);
```

```
if(hasMC) {
  fTree->Branch("Pdg", &pdg);
  fTree->Branch("Pdg_Mother", &pdgmother);
  fTree->Branch("Mother_label", &motherlabel);
  fTree->Branch("Label", &label);
  fTree->Branch("Has_Mother", &hasmother);
  fTree->Branch("IsEnh", &enh);
  fTree->Branch("MCpt", &MCpt);
  fTree->Branch("MCeta", &MCeta);
  fTree->Branch("MCphi", &MCphi);
  fTree->Branch("MCTrack_vtx", &MCvtx);
  fTree->Branch("MCTrack_vtxy", &MCvtxy);
  fTree->Branch("MCTrack_vtxz", &MCvtxz);
}
```

Skimmed trees ([I.Arsene](#))

-used in both Jpsi2ee and LMee-

- See PWGDQ/reducedTree/
- Light framework producing trees from either ESDs or AODs
- Provides:
 - A tree maker analysis task based on an event type structure
 - Analysis tools
 - Data structures: event, track, pair, calo cluster, event plane
 - Cut classes
 - Variable manager and histogram manager
 - Event mixing handler
 - InputEventHandler

Creating the trees

- See [PWGDQ/reducedTree/macros/AddTask_iarsene_dst.C](#)
- Uses [PWGDQ/reducedTree/AliAnalysisTaskReducedTreeMaker.h,.cxx](#)
- Create objects which have an event-like structure ([AliReducedEventInfo](#)) containing:
 - Event wise information
 - List of selected tracks containing
 - Global properties (momentum, charge, quality flags)
 - Specific detector information (ITS, TPC, TOF, TRD, EMCAL/PHOS, MC)
 - List of selected pairs (e.g. V0's or user created candidate pairs)
 - List of calorimeter clusters (EMCAL and PHOS)
 - Detector information: ITS, VZERO, ZDC, TZERO, FMD
 - Event plane
- The current event structure allows for almost any type of analysis
- Flexible tuning of the filtering done in the AddTask by
 - Selection cuts (event, track or pair level)
 - Setting branches as inactive
- The tree maker can be run in either LEGO trains or user analysis
- The trees are typically created a few times (2-3 times) during the analysis

Event data members

TreeViewer

File Edit Run Options Help

Command Option Histogram htemp ☐ Hist ☐ Scan ☒ Rec

Current Folder

- TreeList
- DstTree

Current Tree : DstTree

X)-empty-	fTracks.fTPCDCA[]	fCandidates	fNTracksPerTrackingFlag[]
Y)-empty-	fTracks.fTrackLength	fCandidates.fP[]	fVZEROmult[]
Z)-empty-	fTracks.fMassForTracking	fCandidates.fIsCartesian	fVZEROTotalMult[]
Scissors icon -empty-	fTracks.fChi2TPCConstrainedVsGlobal	fCandidates.fCharge	fZDCnEnergy[]
Box icon Scan box	fTracks.fHelixCenter[]	fCandidates.fFlags	fZDCpEnergy[]
Ex)-empty-	fTracks.fHelixRadius	fCandidates.fQualityFlags	fZDCnTotalEnergy[]
Ex)-empty-	fTracks.fITSclusterMap	fCandidates.fCandidateId	fZDCpTotalEnergy[]
Ex)-empty-	fTracks.fITSSharedClusterMap	fCandidates.fPairType	fTOamplitude[]
Ex)-empty-	fTracks.fITSsignal	fCandidates.fLegIds[]	fTOTOF[]
Ex)-empty-	fTracks.fITSnSig[]	fCandidates.fMass[]	fTOTOFbest[]
Ex)-empty-	fTracks.fITSchi2	fCandidates.fLxy	fTOzVertex
Ex)-empty-	fTracks.fTPCNcls	fCandidates.fPointingAngle	fTOstart
Ex)-empty-	fTracks.fTPCCrossedRows	fCandidates.fChisquare	fTOpileup
Ex)-empty-	fTracks.fTPCNclsF	fEventNumberInFile	fTOsatellite
Ex)-empty-	fTracks.fTPCNclsShared	fL0TriggerInputs	fNCaloClusters
Event	fTracks.fTPCClusterMap	fL1TriggerInputs	fCaloClusters
AliReducedBaseEvent	fTracks.fTPCsignal	fL2TriggerInputs	fCaloClusters.fType
TObject	fTracks.fTPCsignalN	fBC	fCaloClusters.fEnergy
fEventTag	fTracks.fTPCnSig[]	fTimeStamp	fCaloClusters.fTrackDx
fRunNo	fTracks.fTPCchi2	fEventType	fCaloClusters.fTrackDz
fVtx[]	fTracks.fTPCActiveLength	fTriggerMask	fCaloClusters.fM20
fNVtxContributors	fTracks.fTPCGeonLength	fIsPhysicsSelection	fCaloClusters.fM02
fCentrality[]	fTracks.fTOFBeta	fIsSPDPileup	fCaloClusters.fDispersion
fCentQuality	fTracks.fTOFtime	fIsSPDPileupMultBins	fCaloClusters.fPosition[]
fNTracks[]	fTracks.fTOFdx	fIRIntClosestIntMap[]	fCaloClusters.fTOF
fNV0candidates[]	fTracks.fTOFdz	fVtxTPC[]	fCaloClusters.fNCells
fTracks	fTracks.fTOFmismatchProbab	fNVtxTPCContributors	fFMD
fTracks.fP[]	fTracks.fTOFchi2	fVtxSPD[]	fFMD.fMultiplicity
fTracks.fIsCartesian	fTracks.fTOFnSig[]	fNVtxSPDContributors	fFMD.fId
fTracks.fCharge	fTracks.fTOFdeltaBC	fNpileupSPD	fEventPlane
fTracks.fFlags	fTracks.fTRDntracklets[]	fNpileupTracks	fEventPlane.TObject
fTracks.fQualityFlags	fTracks.fTRDpid[]	fNPHDtracks	fEventPlane.fQvector[][][]
fTracks.fTrackId	fTracks.fTRDpidLQ2D[]	fNTRDtracks	fEventPlane.fEventPlaneStatus[][]
fTracks.fStatus	fTracks.fCaloClusterId	fNTRDtracklets	
fTracks.fTPCPhi	fTracks.fMCMom[]	fSPDntracklets	
fTracks.fTPCPt	fTracks.fMCFreezeout[]	fSPDntrackletsEta[]	
fTracks.fTPCEta	fTracks.fMCLabels[]	fSPDFiredChips[]	
fTracks.fMomentumInner	fTracks.fMCPdg[]	fITSClusters[]	
fTracks.fDCA[]	fTracks.fMCGeneratorIndex	fSPDnSingle	

SPIDER STOP

0%

IList OList First entry : 0 Last entry : 432

RESET

Analyzing the reduced trees

- Trees prepared for dielectron analysis are usually very small so it is convenient to download them locally for analysis (on a local farm or even laptop)
- The reducedTree framework allows to run AliAnalysisTask trains on these trees → either locally or on the grid
- Additional filtering of the reduced trees is possible
- Several analyses run by different users; examples:
 - LMee analysis on LHC16l: $\sim 1.0 \times 10^8$ pp events, disk size 10 GB, AOD size ~ 5 TB (Oton)
 - Jpsi2ee analysis on all LHC16d,e,g,h,i,j,k,l,o: 5.1×10^8 MB and 2.5×10^8 HM pp events: disk size ~ 500 GB (Steffen)
 - Jpsi2ee analysis on LHC15o_highIR and LHC15o_pidfix: $\sim 1.0 \times 10^8$ Pb-Pb events, disk size ~ 100 GB (Dennis)
 - Jpsi2ee analysis on LHC15n_pass3, $\sim 1.15 \times 10^8$ MB pp events, 15 GB (Tona)
 - Jpsi2ee analysis on LHC17n_pass1, $\sim 1.4 \times 10^6$ MB Xe-Xe events, 3.2 GB (lonut)
 - General purpose trees on LHC10h and LHC11h, $\sim 6.5 \times 10^7$ kMB, kCentral and kSemiCentral Pb-Pb events, ~ 7 TB, AOD size ~ 70 TB (lonut)

Summary

- Many examples of analyses using data structures from filtered AODs with a very large reduction in data size
 - A big part of the reduction comes from the particular analyses done in DQ
 - But large reduction factors can be obtained with trees even for analysis on hadrons (charged hadrons, identified particles, etc.)
- Tailoring the information to the needs of a specific analysis can bring large advantages: reduction in data size and smaller CPU needs
- Building a framework which allows the user or an analysis working group to easily filter their data would have a large impact