DPD Building with EventView & Future Developments

Amir Farbin
UTA

Overview

- Why build DPDs?
- What is EventView... a reminder.
- Survey of the types of info stored in the DPD.
  - How to create each of these in EventView Framework.
  - How each of these will be represented in the POOL-based DPD created by EV.

Perspective

- AthenaROOTAccess provides direct access to AOD files for fast-turn-around analysis...
  - but AOD files will be too big to use as the input to AthenaROOTAccess on local resources
  - real use-case is creating AOD-like DPDs first... AthenaROOTAccess on DPDs.
- Typical reduction of AOD → DPD size is ~10x (looking at various EV-based AANTs DPDs created by different groups).
  - Mostly thin, slim, and new UserData... little skimming today.
  - The operations required to make the DPD are non-trivial.

Perspective II

- Currently, EV is used in rel-12 to build custom AANT DPDs in all but one Analysis working group.
  - We see that one framework and a common set of extendable tools can meet a wide range of physics use cases.
  - EV Tools and View packages have been rigorously validated through detailed event-by-event comparisons with other code in context of the CSC notes.
    - This has been a lot of work.
    - Do we want to do this every time 2 different people use 2 different analysis code?
- Moving these packages from AANT to POOL-Based DPD should be very easy.

Stages of Analysis

Tier 1/2

- *Re-reconstruction/re-calibration*- often necessary.
- *Algorithmic Analysis*: Data Manipulations ESD→AOD→DPD→DPD
  - *Skimming*- Keep interesting events
  - *Thinning*- Keep interesting objects in events
  - *Slimming*- Keep interesting info in objects
  - *Synthesis*- Build new data structures from building blocks. Encapsulate the results of algorithms.
  - Basic principle: Smaller data → more portable & faster

Focus on Thinning and Synthesis today... the others are easy.

Tier 3

- *Interactive Analysis*: Making plots/performing studies on highly reduced data.
- *Statistical Analysis*: Perform fits, produce toy Monte Carlos, calculate significance.

Stages of Analysis

- Use TAG to quickly select subset of events which are interesting for analysis.
- Starting from the AOD

Physics Group

- Stage 0: Re-reconstruction, re-calibration, selection (AOD)
  - Redo some clustering/track fitting, calculate shower shapes, apply corrections, etc...
  - Typical: 250 ms/event, In: 75% AOD, out 50% AOD

Analysis Group

- Stage 1: Selection/Overlap removal/complicated analysis (AOD/DPD)
  - Select electrons/photons→find jets on remaining clusters→b-tag→calculate MET
  - Perform observable calculation, combinatorics + kinematic fitting, ...
  - Typical: 20 ms/event, In: 25% AOD, Out: 10% AOD

Personal

- Stage 2: Interactive analysis (AOD/DPD)
  - Final selections, plots, studies.
  - Prototype earlier steps!
  - Typical: 0 ms/event, In: 1% AOD, Out: 0
- Stage 3: Statistical Analysis

Stages vs Resources

- ATLAS will record 200 Hz of data, regardless of luminosity ➔ 10^9 event/year.
- CM Assumption 700 Analyzers: 12 tier 2 CPU/person for analysis at any give time.
- Assuming perfect software/hardware (10 MB/s read in = ROOT limit).

		Laptop 1 Cores	Tier 3 25 Cores	Tier 2 10 Persons 100 Cores	Tier 2 100 Persons 1000 Cores	
Step 0	1 Hour	0.0001%	0.0035%	0.0140%	0.1398%	Working group on Tier 2
	Overnight	0.0017%	0.0419%	0.1678%	1.6777%	
	1 Week	0.0235%	0.5872%	2.3487%	23.4874%	
	1 Month	0.1007%	2.5165%	10.0660%	All	
Step 1	1 Hour	0.0016%	0.0400%	0.1600%	1.6000%	Analysis group on Tier 2
	Overnight	0.0192%	0.4800%	1.9200%	19.2000%	
	1 Week	0.2688%	6.7200%	26.8800%	All	
	1 Month	1.1520%	28.8000%	All	All	
Step 2	1 Hour	0.3600%	9.0000%	36.0000%	All	Single Analyzer on Tier 3
	Overnight	4.3200%	All	All	All	
	1 Week	60.4800%	All	All	All	
	1 Month	All	All	All	All	

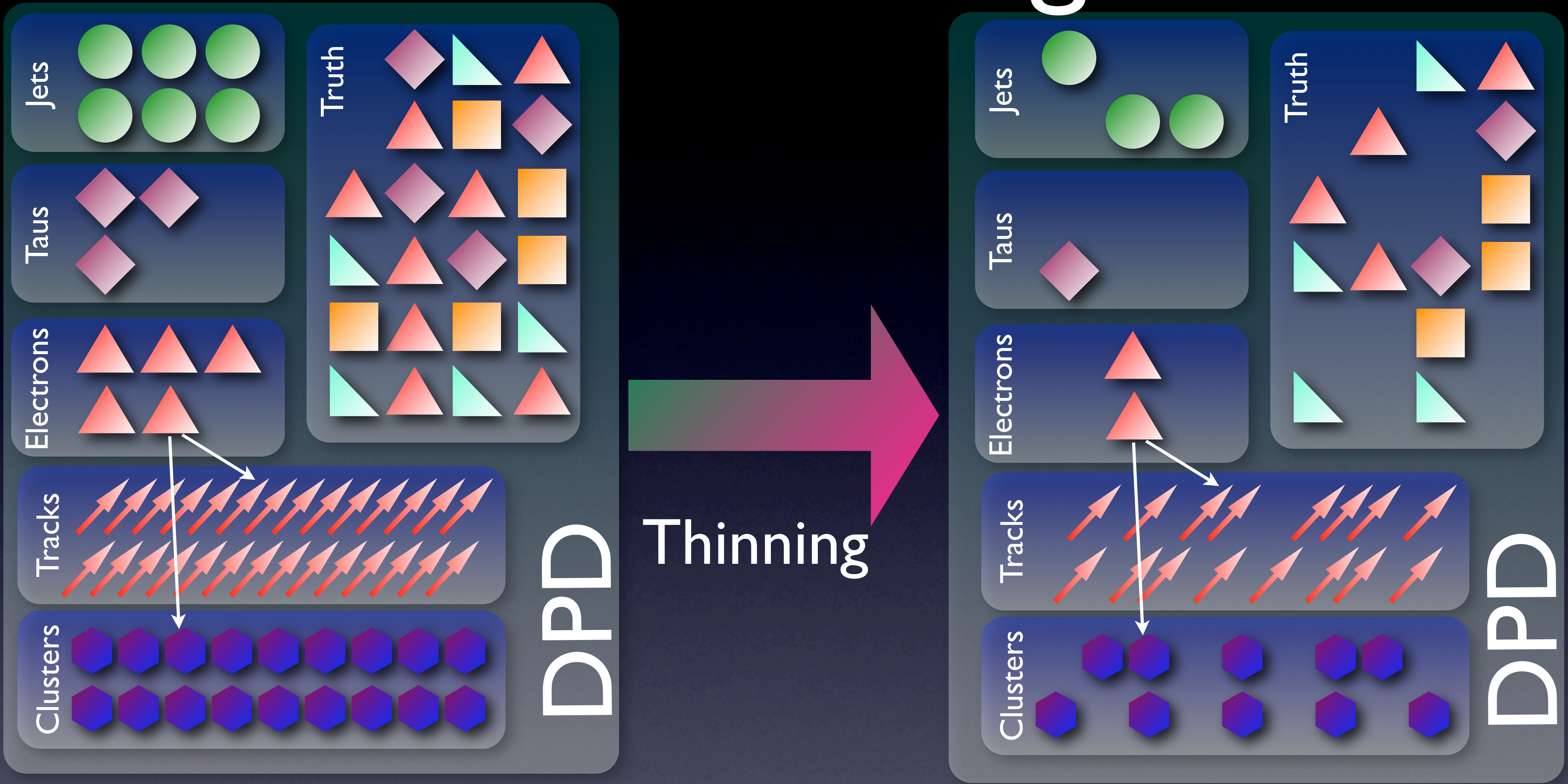
Stages vs Resources

- For LHC data value it is necessary to do analysis in stages and make DPDs, because:
  - The full AOD for skimmed samples are too large to fit on local resources (eg laptop or even tier 3). (1 TB= 10 M full AOD events or 1% of 1 year's AOD)
  - It would take too long to do all steps of an analysis every time you want to make a new plot or change a cut.
- Even making simple plots which require no processing on 1 KB-per-event DPDs will take hours to go through reasonable amounts of data..
- Sophisticated processing of events can easily dominate over IO (see my talk at the 2006 Analysis Model Workshop).
  - Ex: Full top analysis optimistically takes ~20 ms/event. This is equivalent to 200 KB/s AOD read-in.
- Your ROOT macro analysis of CSC data will not scale to LHC data-volumes.

Quick Reminder

- EventView/UserData are data (EDM) objects which
  - keep track of what other EDM objects used in analysis.
  - store information beyond what is in the AOD/ESD.
  - Present a much nicer C++ interface than bare Athena, StoreGate, containers, and EDM objects.
- EventView Framework allows building analysis from modular pieces.
  - An analysis is a chain of tools, which can be developed independently but will work together because of EventView.
  - Full flexibility of any Athena-based analysis... + more.
- EventViewBuilder Library is a large collection of generic tools which are chained and configured in python (like reconstruction or anything else in athena).
  - Users don't need to write C++ to quickly get a lot of functionality.
  - Defines basic concepts (eg Insertion, Looping, Association, ...) which can be extended by users.
- View Packages are collection of EVTools and configuration which produce standard DPDs.
  - Serve nearly all physics and performance groups.
- All of this makes EventView a very powerful and widely-used DPD making framework.

DPD Building

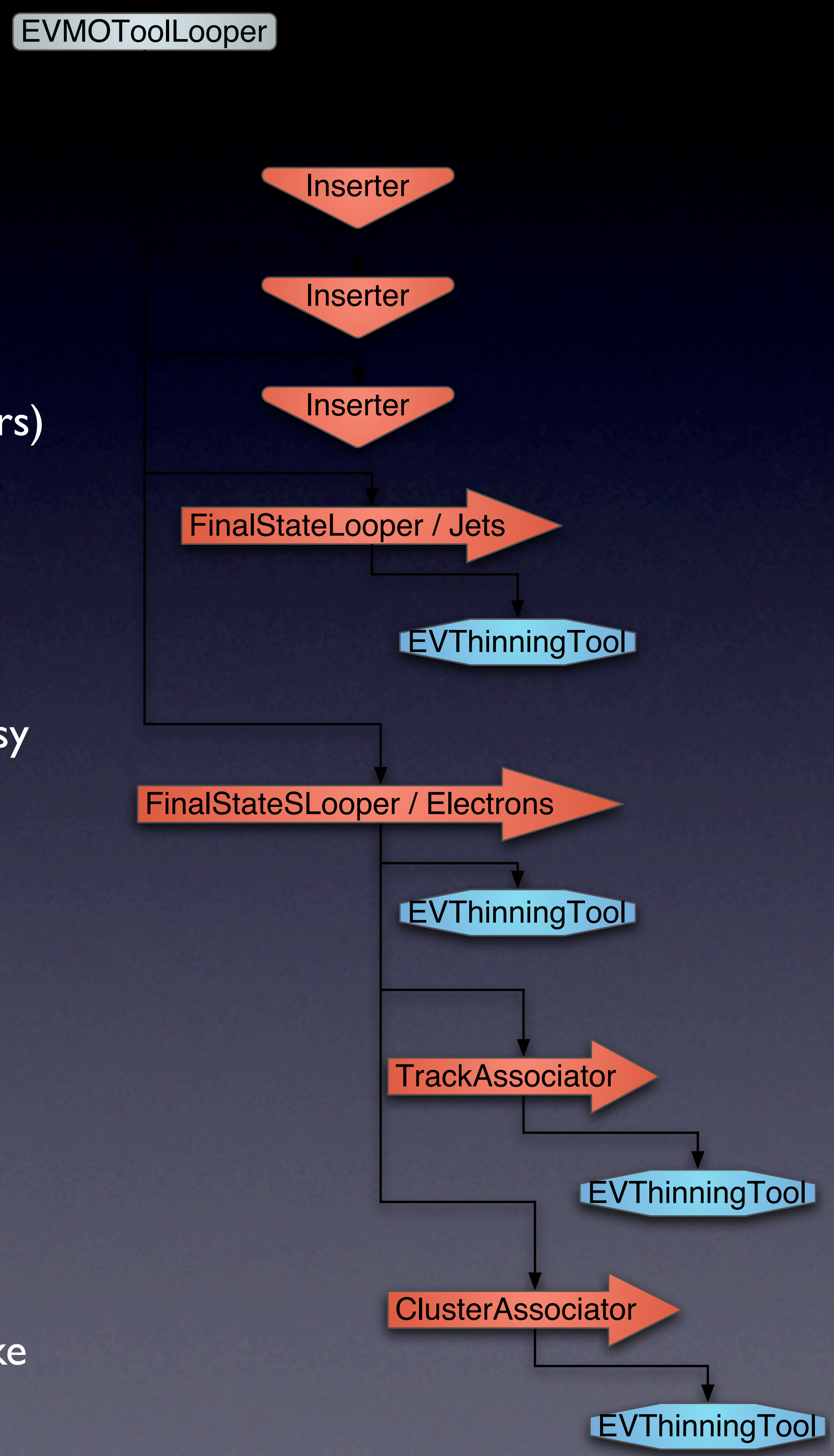


Example

- Keep the "good" electrons
  - their associated tracks and clusters
  - tracks and clusters in a cone
- Keep truth electrons coming from W, Z's, or SUSY particles.

How to thin?

- The thinning service simply allows users to mark objects they want written out... does not make any decisions.
- What you need to thin (not EV specific):
  - Select "seed" objects. (Requires physics input)
  - Mark them to be saved.
  - Follow each seed object and mark their constituents (ie tracks & clusters) to be saved.
  - Find objects in cone around the seed, mark them to be saved. (Requires physics input + matching)
- EV tools have been doing this type of operation for almost 2 years now... Easy to do, documented in tutorials, requires only job options. Lots of people in ATLAS use these mechanisms.
- Since the physics decisions and DPD technology are separated in EV... all we needed to do was add one new tool to allow EV to thin and create POOL-based DPD (Kyle has put it into 13.0.30).
- Allows completely standardization of the thinning process. This works for everything (reco, truth, fastsim, trigger).
- BTW, this means that current EV analyses should be easily updated to the POOL-based DPD format (and if you like, simultaneously create old ones, like AANT).



Thinning/Slimming with HighPtView

- HighPtView is fully configurable via external options.
- Don't need to expertise on Athena, EventView, etc...
- Just follow instructions on how set these options.
- This is DPD making for AthenaROOTAccess made easy and completely standardized.
- Lots of people already use this mechanism to build custom DPDs with EV.
- Ex: SUSY group has 7 different HPTV/SV based DPDs.

MyOptions.py

```
InserterConfiguration={ "Electron":
                        {"FullReco":
                         [
                          {"Name":"ElMedium", "Configuration":{"etCut":10*GeV}},
                          {"Name":"ElLoose",  "Configuration":{"etCut":10*GeV}},
                          {"Name":"ElTight",  "Configuration":{"etCut":10*GeV}}]} ,
                        "Muon":
                        {"FullReco,Muid":
                         [ {"Name":"MuSUSY",
                            "Configuration":
                            { "ContainerKey" :"MuidMuonCollection",
                              "etCut" : 15*GeV,
                              "onlyHighPt" : False,
                              "relativeIsolationCut" : 1.,
                              "useChi2FromCombinedMuon": True,
                              "chi2NdofCut":5,
                              "chi2MatchCut":20,
                              "deltaRCut":.1,
                              "RemoveOverlapWithSameType" : False,
                              "InsertedLabels":["Muid", "Muon","Lepton"],
                              }},
                           {"Name":"MuDefault"}]}}

# Some of this would be replaced by
DetailLevel= ["POOLDPD", "El_Info:ClusterInfo", "Ph_Info:ClusterInfo",
"Ph_Info:TracksInCone"]

DPDOutputList= ["ElectronContainer", "MuidMuonCollection", "ElectronShowerEgDetail",
"TrackParticleCollection",...]
```

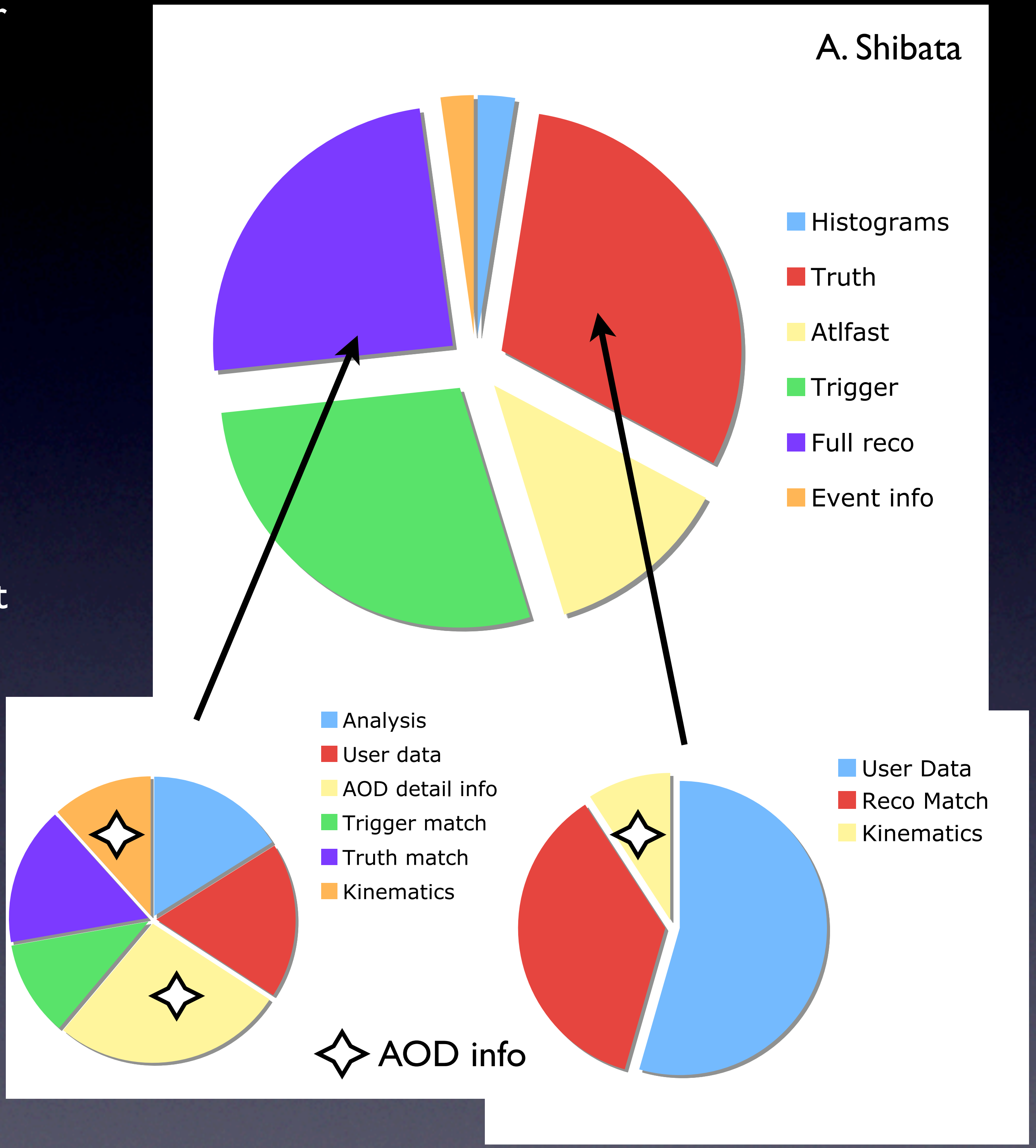
```
pathena MyOptions.py HighPtView/HighPtViewDPD_topOptions.py --inDS
csc12.005406.SU8_jimmy_susy1.recon.AOD.v130030 --outDS MyDPD
```

DPD Contents

- Two types of DPD:
  - "Performance" DPDs: subset of information/events necessary for calibrations and performance studies. For early data or group wide DPD. Necessary to speed up iterations and/or use local resources.
  - "Analysis" DPDs: Tailored to specific analysis and user preferences.
- Two categories of information:
  - Information originally in the AOD (possibly re-reco'ed, re-calibrated, or corrected):
    - Ex: Tight/Medium Electrons, their tracks and clusters, and every track within cone 0.1 around them and the closest topo-cluster.
    - All true Electrons which come from a t->Wb->e nu jet chain.
  - Information not in AOD, often referred to as UserData: (Example)
    - "Labels": The fact that the electron is Tight or Medium, it was used in W reco... Flags that the true electron was reco'ed as Jet or Tau... that the true electron came from a W...
    - The association between the Electron and the tracks/clusters around it.
    - The association between the true, reco, trigger Electron.
    - Composites Objects (or just their kinematics)
    - Event Shape Variables etc...

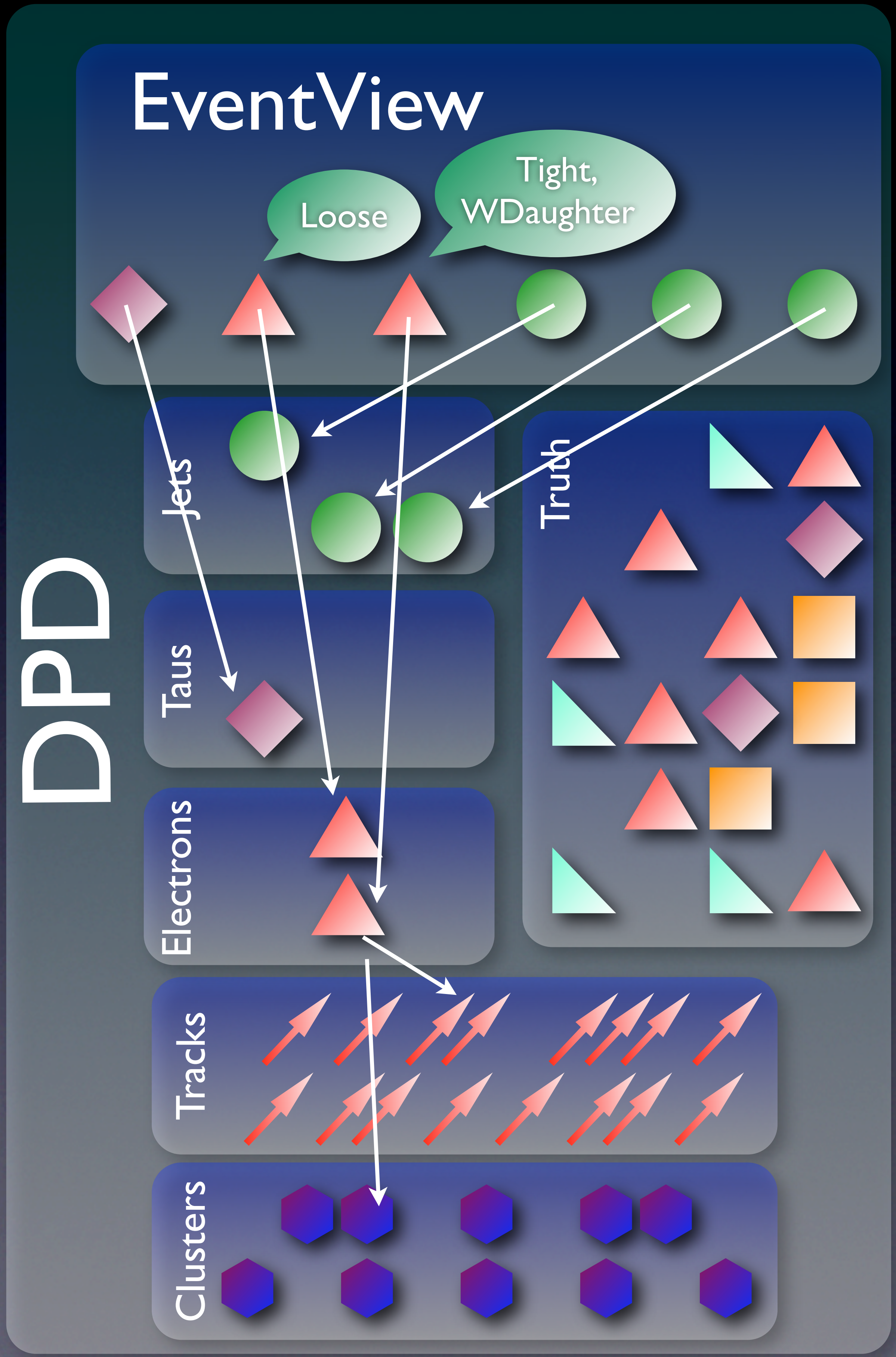
“UserData”

- Yesterday we saw that many groups had added to their ntuples information which was not originally in the AOD.
  - Object quality info (eg Tight Electron flag)
  - Matching info (Truth Match, Trigger Match)
  - Event Quantities (sphericity)
  - etc...
- Many of these are calculable on the DPD in ROOT, but
  - often one double (per object?) is all you need in the rest of the analysis, so you can reduce DPD size by not saving the inputs to the calculation.
  - you can save a lot of ROOT processing time by caching the result in the DPD.
  - often very convenient to have these quantities pre-calculated.
    - With HighPtView and most other EV-based DPD you can make efficiency, resolution, scale plots for any reco or trigger object with single-line ROOT commands.



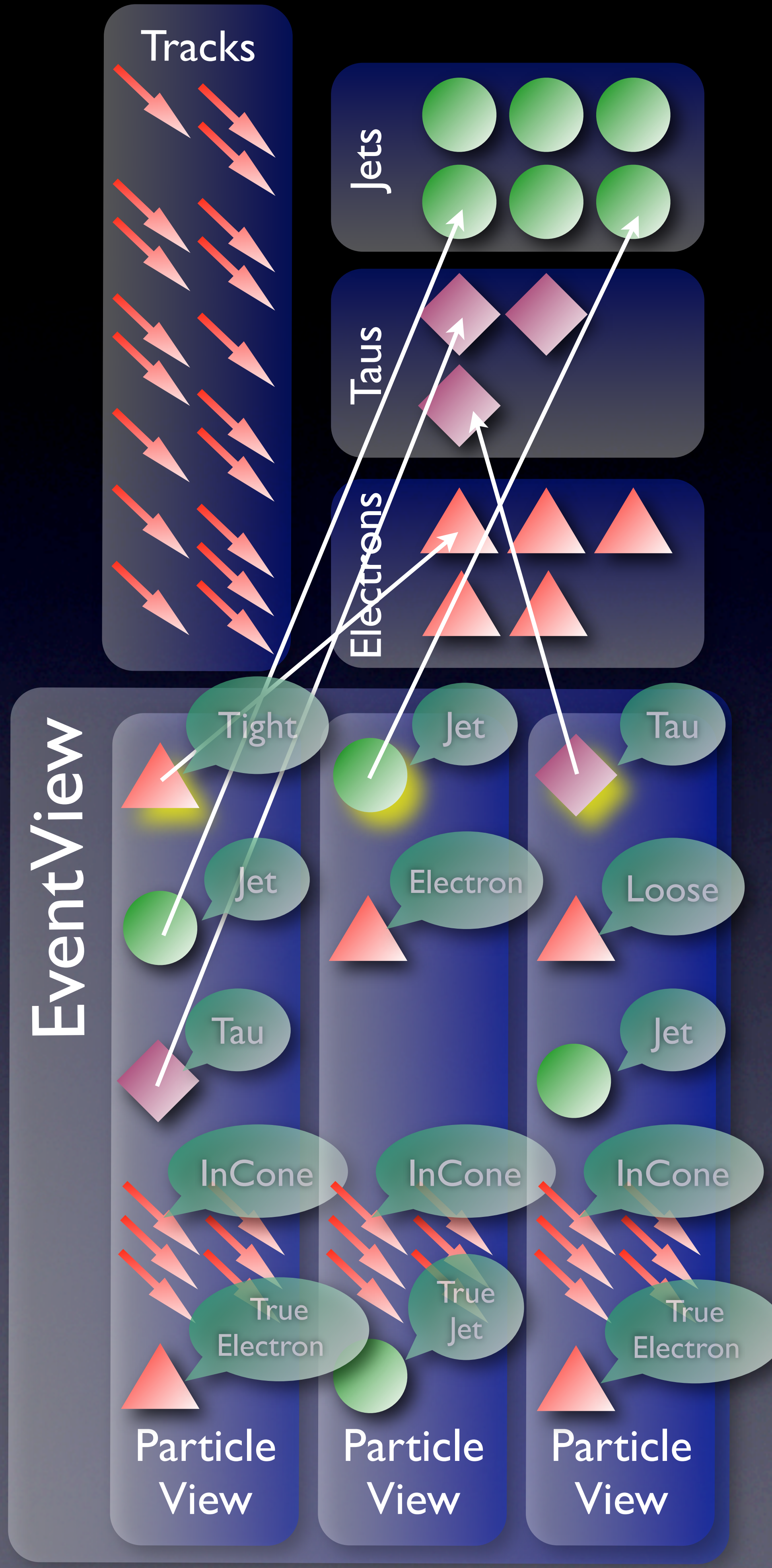
EventView is “UserData”

- How can you tell why an object is in the DPD?
- EventView keeps pointers to the objects in the AOD, along with labels.
- Provides simple interface to get objects:
  - `std::vector<const Electron*> *TightElectrons = ev->finalStateObjects<Electron>(“Tight”);`
  - `std::vector<const INavigable4Momentum*> *Everything = ev -> finalStateObjects<INavigable4Momentum>();`
  - `bool isTight = ev->hasLabel(obj,”Tight”);`
  - `std::vector<std::string> ElLabels = ev -> labelsFor(obj);`
- EventView can be written by POOL into the DPD along with the AOD objects. (Needs T/P separation for speed optimization).
- EventView should be accessible through AthenaROOTAccess, just like anything else. (Needs work/testing).



ParticleView

- One of the concerns with EV has been the inability to adjust overlap removal during DPD analysis.
- Idea:
  - Use EventView framework to figure out what objects overlap.
  - Save result in ParticleView/EventView.
- ParticleView (being developed by Peter Sherwood):
  - Keeps pointers to several "interpretations" of a "physics" object.
  - It is a particle (ie has 4 momentum). Presents the kinematics of the "active" interpretation.
  - Allows switching active interpretation.
  - Similar nice interface as EventView... and accessible in AthenaROOTAccess.
- ParticleView can also keep track of associations... like truth, trigger matched objects, or tracks in cone.

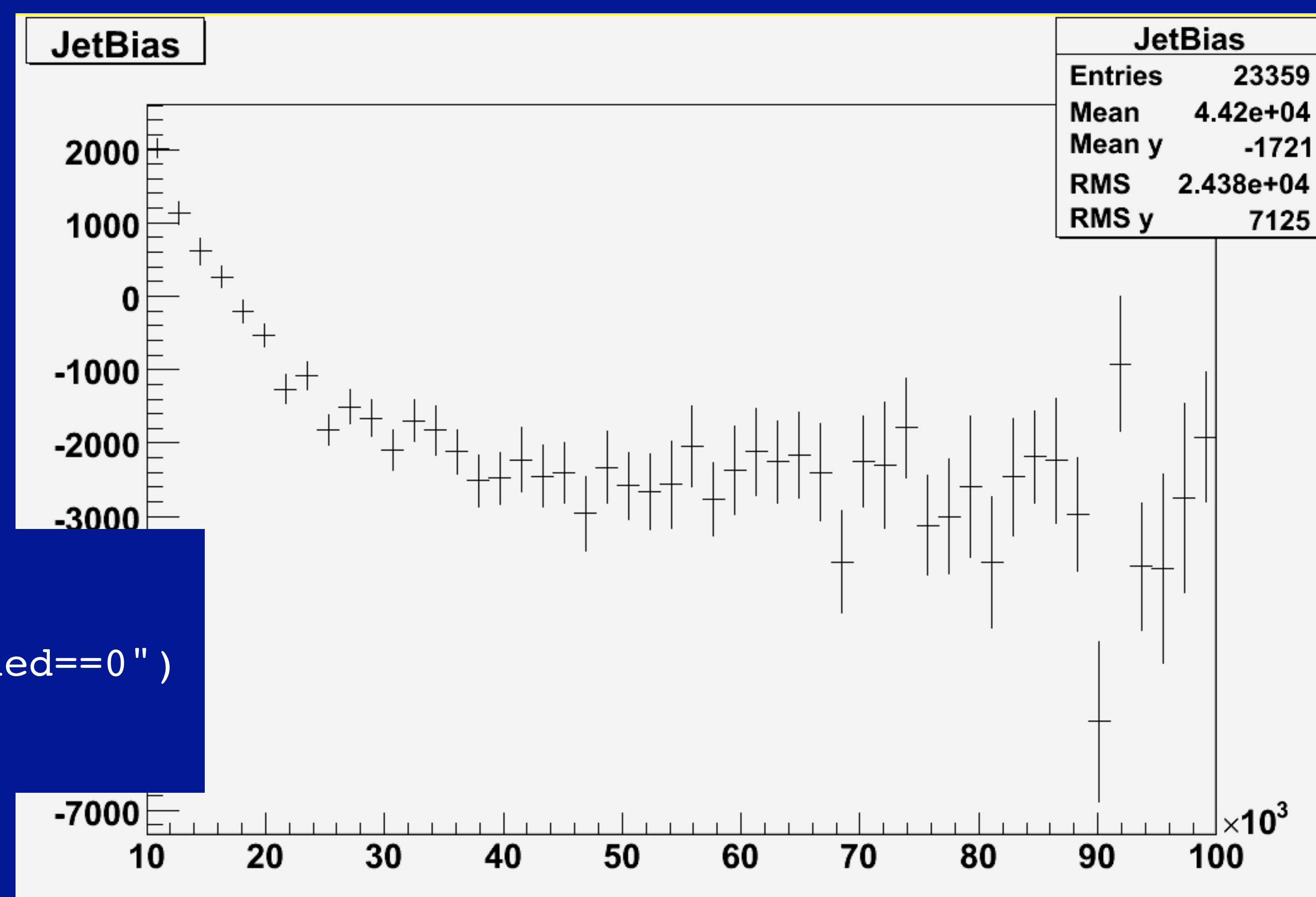


More Powerful DPDs

```
Truth0->Scan("El_p_T:El_eta:El_R_p_T:El_etcone","El_N>0&&El_p_T>15000")
************************************************************************
*    Row   * Instance *    El_p_T *    El_eta *  El_R_p_T * El_etcone *
************************************************************************
*       18 *        0 * 105285.32 * 0.4091242 * 103055.27 *  510.9823 *
*       28 *        0 * 25344.389 * -0.613313 *         0 * 1010.6964 *
*       31 *        0 * 41348.116 * -2.036790 *         0 * 567.92218 *
*       42 *        0 * 55272.240 * -0.556607 * 52544.099 *         0 *
*       74 *        0 * 22167.001 * 0.4038631 * 21070.964 *         0 *
*       77 *        0 * 24399.744 * -1.552361 * 24093.357 * 1868.9583 *
*       79 *        0 * 48076.995 * 1.5331588 *         0 * 81.795562 *
*       90 *        0 * 117659.36 * -0.131803 *         0 * 1645.3580 *
*       90 *        1 * 17144.797 * 1.2736656 *         0 *         0 *
*       93 *        0 * 54085.071 * 0.7382863 * 53275
*       93 *        1 * 32654.367 * 1.0285901 * 30818
*      102 *        0 * 41567.572 * 1.7180224 * 40147
*      126 *        0 * 19774.637 * 2.1294892 * 19313
*      131 *        0 * 98802.467 * 0.3206566 * 94770
*      134 *        0 * 100685.01 * -0.191601 * 99041
*      134 *        1 * 178483.36 * -0.258732 * 17452
*      159 *        0 * 47150.007 * -0.378147 * 46348
*      161 *        0 * 131364.69 * 0.8869048 * 13121
```

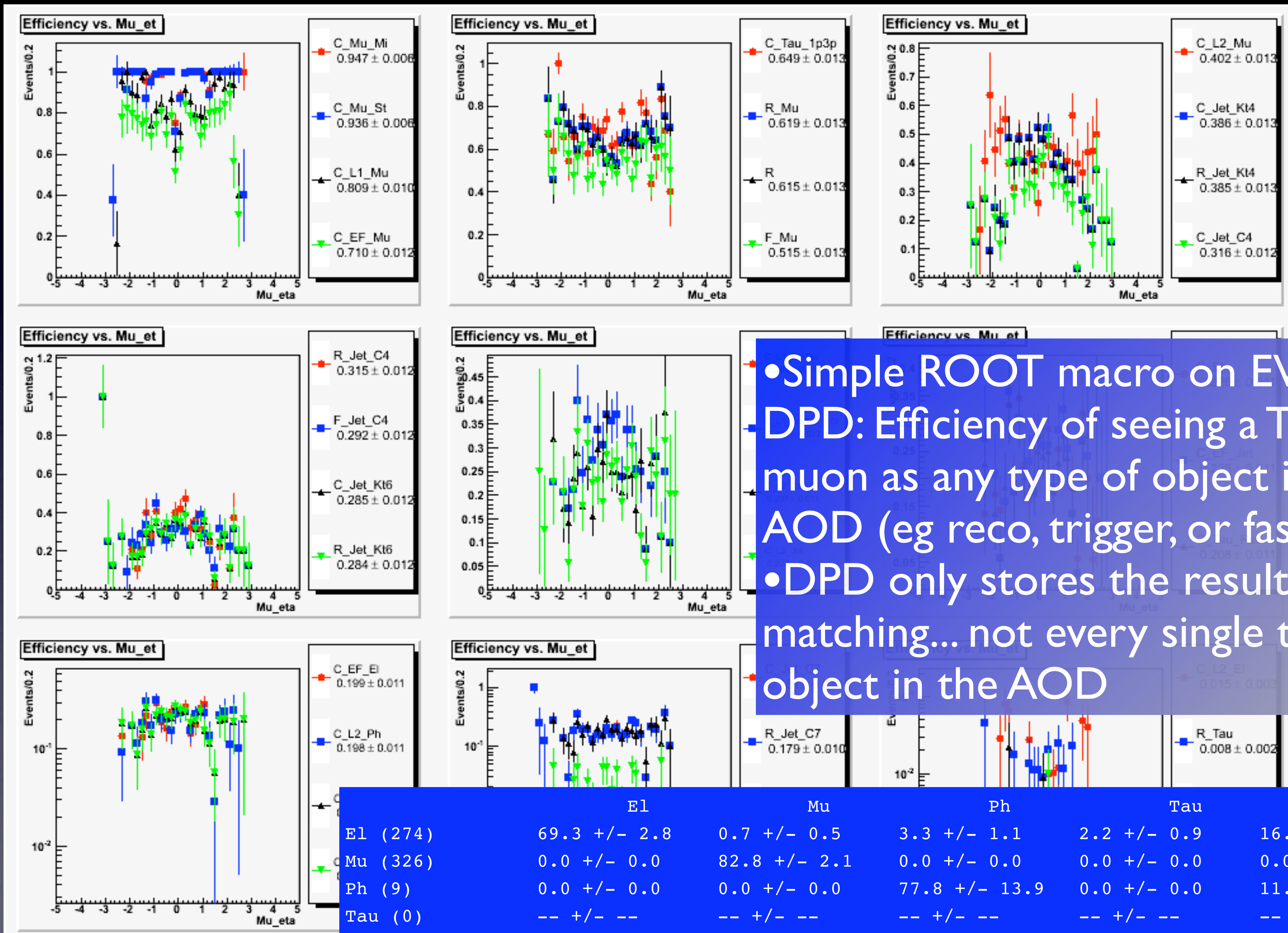
- DPD made with EV allow making efficiency, resolution, scale, etc plots on the ROOT prompt.
- AANT-based HPTV AANT shown here... Same will be true of POOL-based DPD.

```
TProfile JetRes("JetBias","JetBias",50,10000,100000)
FullRec0->Draw("Jet_C4_p_T-Jet_C4_T_p_T:Jet_C4_T_p_T>>JetBias",
"Jet_C4_N>0&&Jet_C4_T_Matched==1")
```




```
FullRec0->Draw("El_p_T","El_N>0")
(Long64_t)697
FullRec0->Draw("El_p_T","El_N>0&&El_T_Matched==0")
(Long64_t)49
```

Global View



- Simple ROOT macro on EV-made DPD: Efficiency of seeing a True muon as any type of object in AOD (eg reco, trigger, or fast sim)
- DPD only stores the result of matching... not every single type of object in the AOD

	El	Mu	Ph	Tau	Jet
El (274)	69.3 +/- 2.8	0.7 +/- 0.5	3.3 +/- 1.1	2.2 +/- 0.9	16.8 +/- 2.3
Mu (326)	0.0 +/- 0.0	82.8 +/- 2.1	0.0 +/- 0.0	0.0 +/- 0.0	0.0 +/- 0.0
Ph (9)	0.0 +/- 0.0	0.0 +/- 0.0	77.8 +/- 13.9	0.0 +/- 0.0	11.1 +/- 10.5
Tau (0)	-- +/- --	-- +/- --	-- +/- --	-- +/- --	-- +/- --
Jet (7335)	0.5 +/- 0.1	0.4 +/- 0.1	0.2 +/- 0.1	0.7 +/- 0.1	47.6 +/- 0.6

Other "UserData"?

- Imagine that for every electron you would like to save into the DPD
  - The distance to the closest Track and Jet, but not the actual Track and Jet.
  - The result of the newest MVA electron discriminant (eg Boosted-decision tree), but not all of the inputs which go into it.
  - a "correction" factor calculated from parameters in the database and/or geometry info.
- And for the event:
  - The energy in topo-clusters which don't overlap with any selected/overlap-removed high-p_T object.
  - STransverse Mass (or similar observable) which is time consuming to repeatedly recalculate and requires selection/overlap-removal as a prerequisite.

Storing "UserData"?

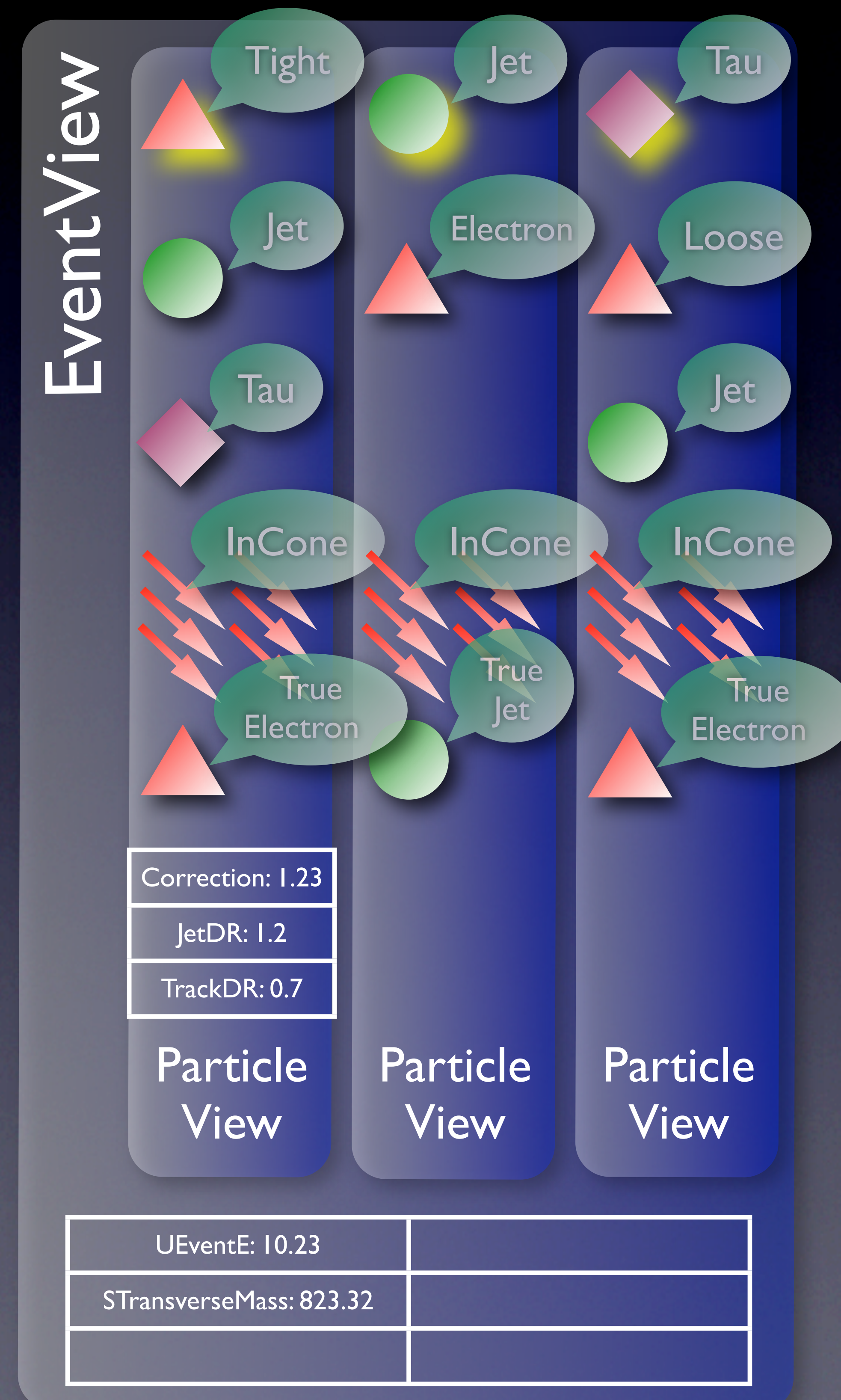
- How do you such put infomation into the new DPD and read it back in AthenaROOTAccess?
  - You can make a TTree (ie AANT) which you save in the POOL file.
    - Only one client (ie Algorithm/Tool) can simultaneously access a given branch in a TTree.
      - Difficult to modularize your code.
    - No reliable mechanism to link the observable to an object, eg the Electron correction to the Electron object or the STransverse mass to the objects used in the minimization.
- You can create a new Athena EDM object and write it out with POOL
  - Requires expertise to implement the object, persistify with POOL, read back in AthenaROOTAccess, worry about schema evolution.... just to put a single double into the DPD.

The UserData EDM Object

- The UserDataBlock EDM object is a better solution.
- Create it:
  - UserDataBlock *myUD=new UserDataBlock();
- Fill it in Athena (with any type):
  - sc=myUD->put<int>(“Blah”,10.);
  - sc=myUD->put<vector<double> >(“BlahVec”,vec);
- Store it into SG and tell POOL to write it out:
  - sc=sgsvc->record<UserDataBlock>(“MyUserData”,myUD);
  - DPDOutputList+=[“UserData#MyUserData”] # In job option
- Read it from SG and get values (in same job or on the DPD in Athena or AthenaROOTAccess):
  - sc=myUD->get<vector<double> >(“BlahVec”,vec);
- Read it from the DPD file in pure ROOT (even without AthenaROOTAccess)
  - MyUserData->Draw(“BlahVec”);
- Note, though the Athena side of UserData works and many of you have been using it, we need to develop the DPD side.

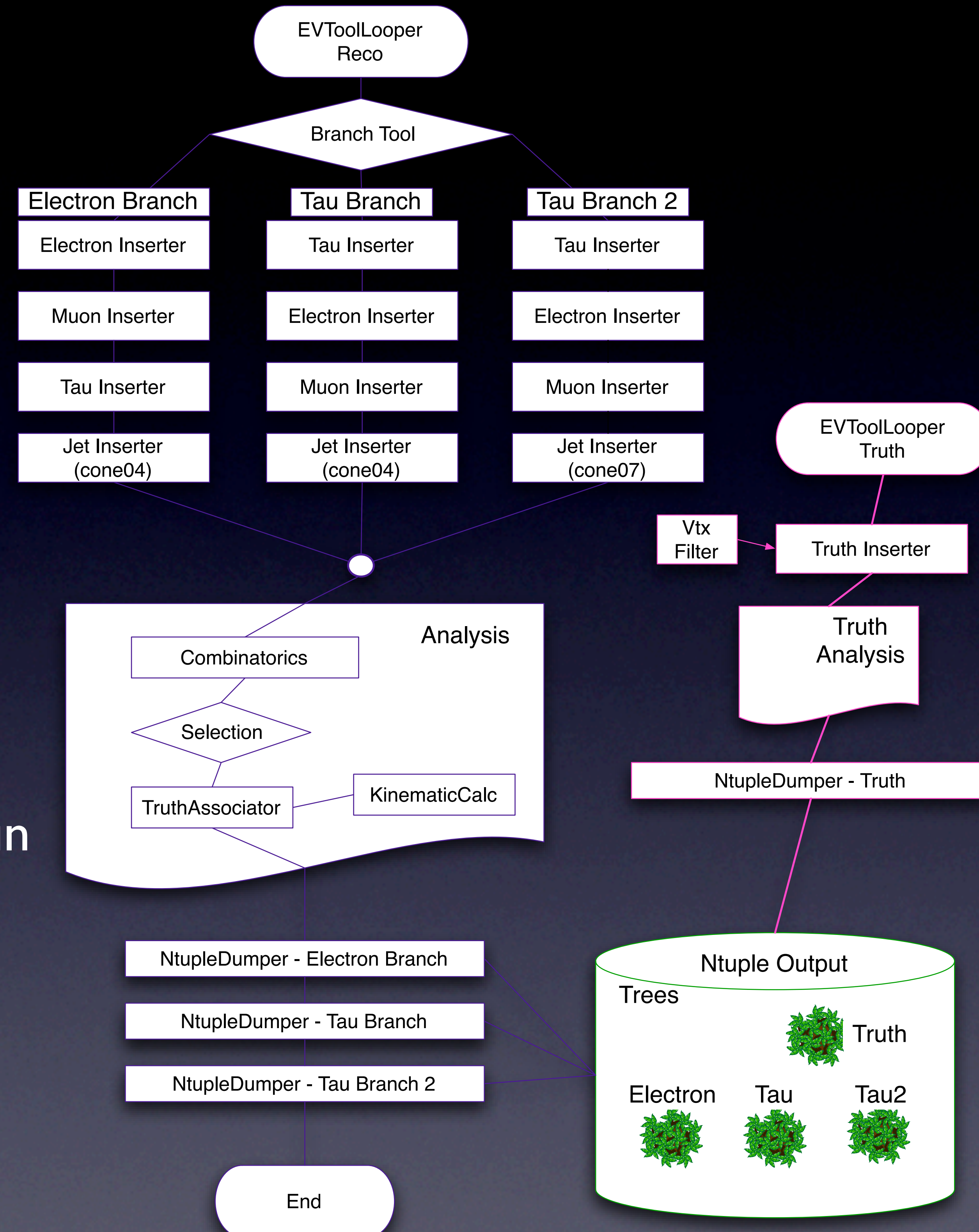
UserData and EventView

- EventView and ParticleView hold instances of UserData.
- Easy to fill/retrieve:
  - ev->userData<double>(“UEventE”,x);
  - pv->userData<double>(“Correction”,y);
- Natural book keeping:
  - Keeps event quantities with the Event
  - Keeps particle quantities with the particle.



Book-keeping

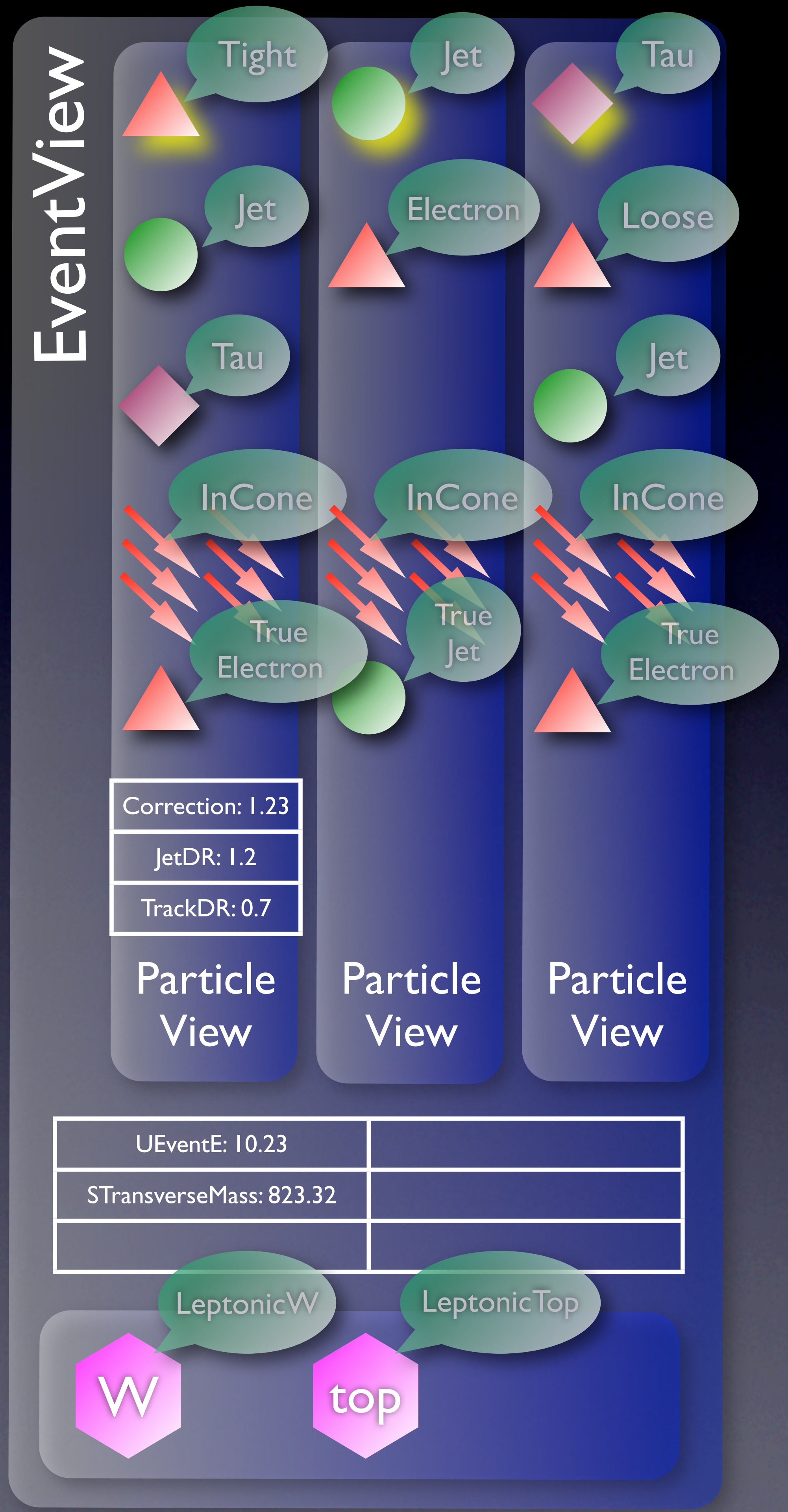
- Recall: EV Analysis are typically a series of tools run in order.
- Because EV is so general, we commonly apply the exact same analysis to Reco, FastSim, Truth, and even Trigger objects.
- Branch tool allows you to simultaneously consider different approaches.
- Powerful means of studying systematics.
- With a few lines of job Option, you can run the same analysis several times with different Jet, Muon, or Tau Algorithms, selections/overlap removal, etc.
- Each branch results in it's own self-consistent EventView(s).
- Then you can event-by-event compare results on your DPD.



Create Write POOL-based DPD instead of AANT

EV in the DPD

- Composite Particles are also a form of "UserData".
- The EventView/ParticleView annotate the DPD
  - Keep pointer to all objects used in analysis
  - Use labels to keep the track of every object's role in the analysis.
  - Preserve the relations between objects (eg reco ↔ truth)
  - Store any user generated data and keep the relation with the particles, events, and analysis branch.

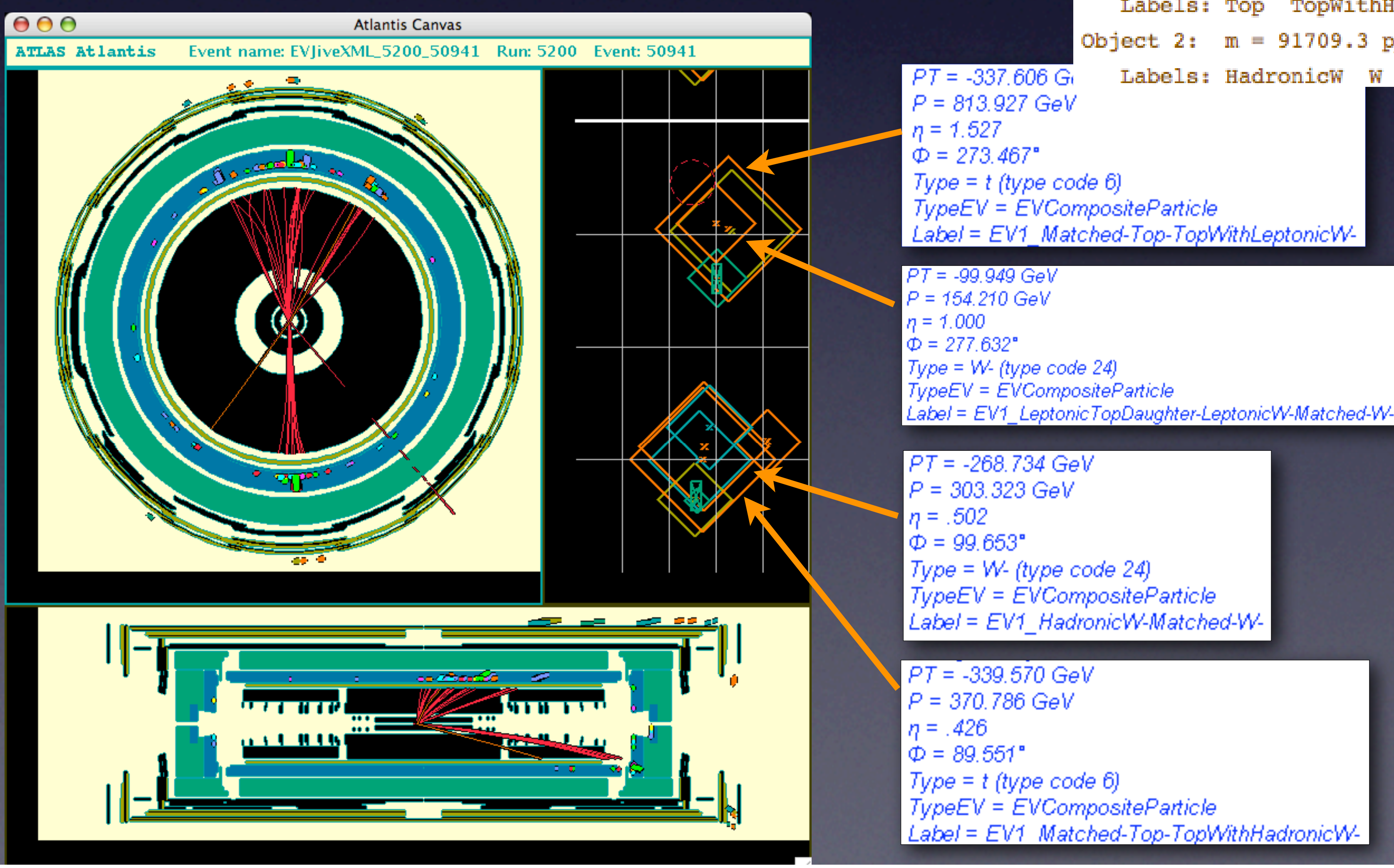


EV in the DPD

- EV stores all of the results of any EV analysis in a format that is common to all analyses... regardless of what was done in the analysis.

ScreenDump from Athena

```
---------------- Final State Objects ------
Object 0:  p_T = 55461.3 phi = -2.18794 eta = 1.04014 type = Analysis::Muon
   Labels: Lepton  Muid  Muon  Tight
Object 1:  p_T = 10807.9 phi = 0.978227 eta = 0.17026 type = Analysis::Muon
   Labels: Lepton  Loose  Muid  Muon  Tight
Object 2:  p_T = 238032 phi = -1.54079 eta = 1.69019 type = ParticleJet
   Labels: CentralJet  Cone4  HardJet  ParticleJet
Object 3:  p_T = 160690 phi = 1.54603 eta = 0.308174 type = ParticleJet
   Labels: CentralJet  Cone4  HadronicTopDaughter  HadronicWDaughter  HardJet  ParticleJet
Object 4:  p_T = 115243 phi = 2.01035 eta = 0.720166 type = ParticleJet
   Labels: CentralJet  Cone4  HadronicTopDaughter  HadronicWDaughter  HardJet  ParticleJet
Object 5:  p_T = 88584.7 phi = 1.00186 eta = 0.0929353 type = ParticleJet
   Labels: BTagged  CentralJet  Cone4  HadronicTopDaughter  HardJet  ParticleJet
---------------- Inferred Objects ------------------
Object 0:  m = 814805 p_T = 67360 phi = 1.81452 eta = 3.22118 type = CompositeParticle
   Labels: AllObjVectSum
Object 1:  m = 168641 p_T = 339570 phi = 1.56295 eta = 0.425561 type = CompositeParticle
   Labels: Top  TopWithHadronicW
Object 2:  m = 91709.3 p_T = 268734 phi = 1.73927 eta = 0.502074 type = CompositeParticle
   Labels: HadronicW  W
```



- You can open someone else's POOL-based DPD, print the EVs and look at them in a Atlantis.
- You can read in the EVs in Athena and continue where the previous step left off.

Evolution of EventView

New Features

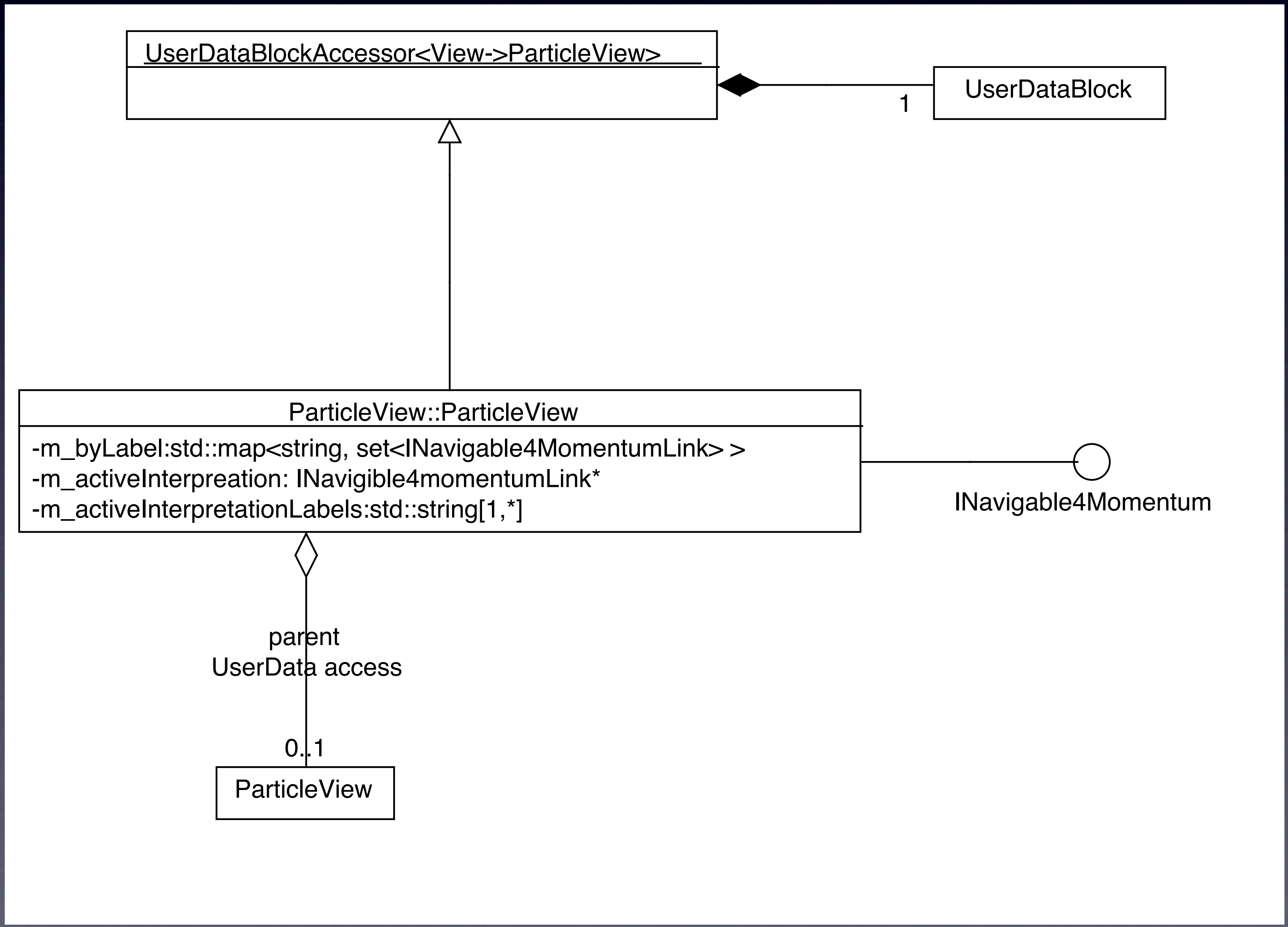
- Thanks to Liza, the auto-generated Inserter Cut-flow table now retain cut order... and look nicer.
- In development: automatic validation histograms
    - histogram efficiency of every cut wrt to user-defined variables (p_T, eta, phi, isolation, ...)
    - this is information which is lost during DPD making.

```
FullRecoLooperMuidTau1p3p.InsertersMuidTau1p3p_Electron_HighPtInser...   INFO
CUT RESULTS: ElectronCollection
==============================================================================
= Cut                            Num Passed     Cut Effic.     Cut Flow Eff.  =
=----------------------------------------------------------------------------=
= All                            1374           1              1              =
= ptCut                          1374           1              1              =
= etCut                          425            0.309          0.309          =
= eCut                           425            1              0.309          =
= authorCut                      425            1              0.309          =
= isolationCut                   425            1              0.309          =
= caloCut                        33             0.0776         0.024          =
= track Quality Cut              19             0.576          0.0138         =
= All_Preselection               19             1              0.0138         =
= Overlap                        19             1              0.0138         =
= Inserted                       19             1              0.0138         =
==============================================================================
```

ParticleView Implementation

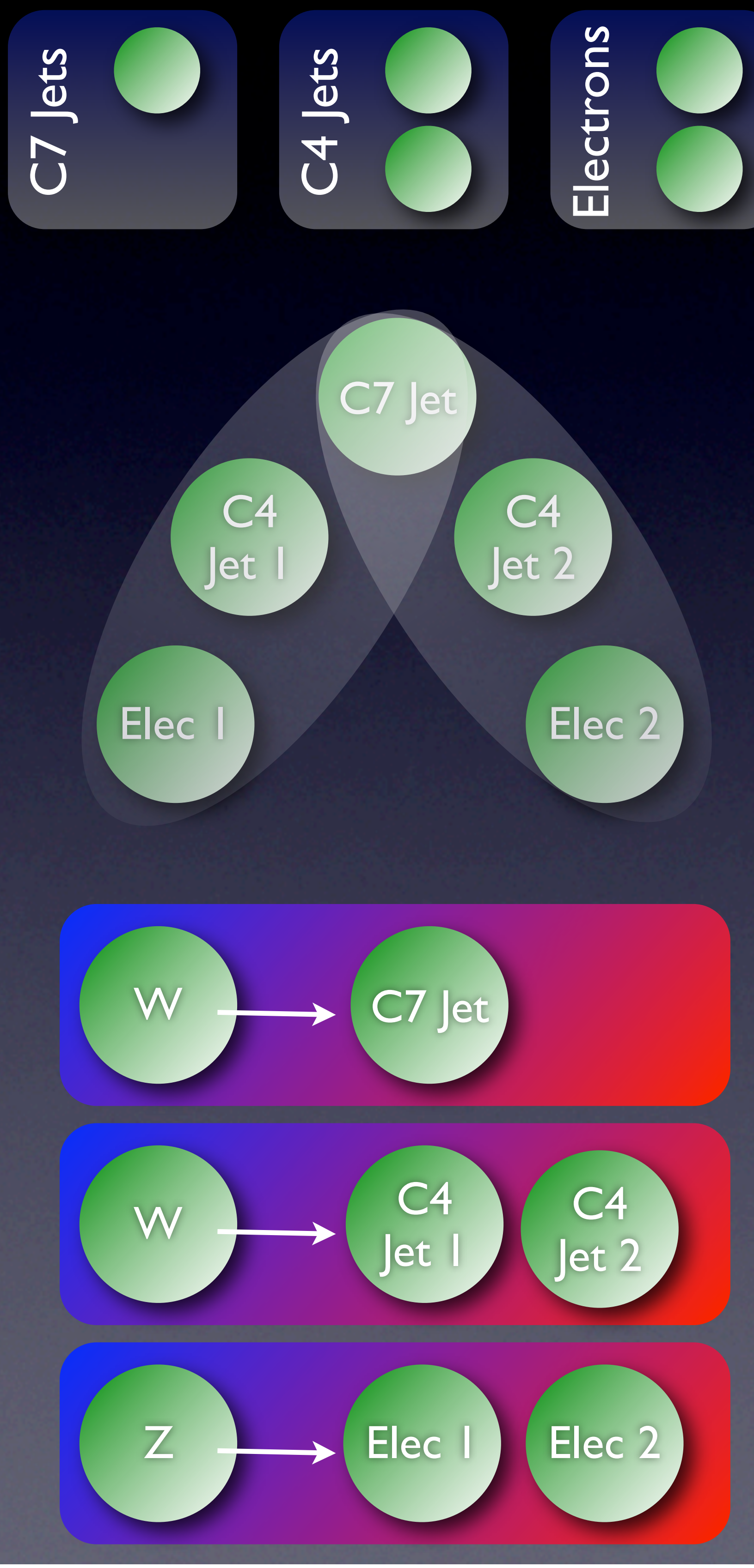
Peter Sherwood

- Interface/ Implementation is similar to EventView
- UserDataBlockAccessor base class provides the UserData interface... will likely be used for EventView also.

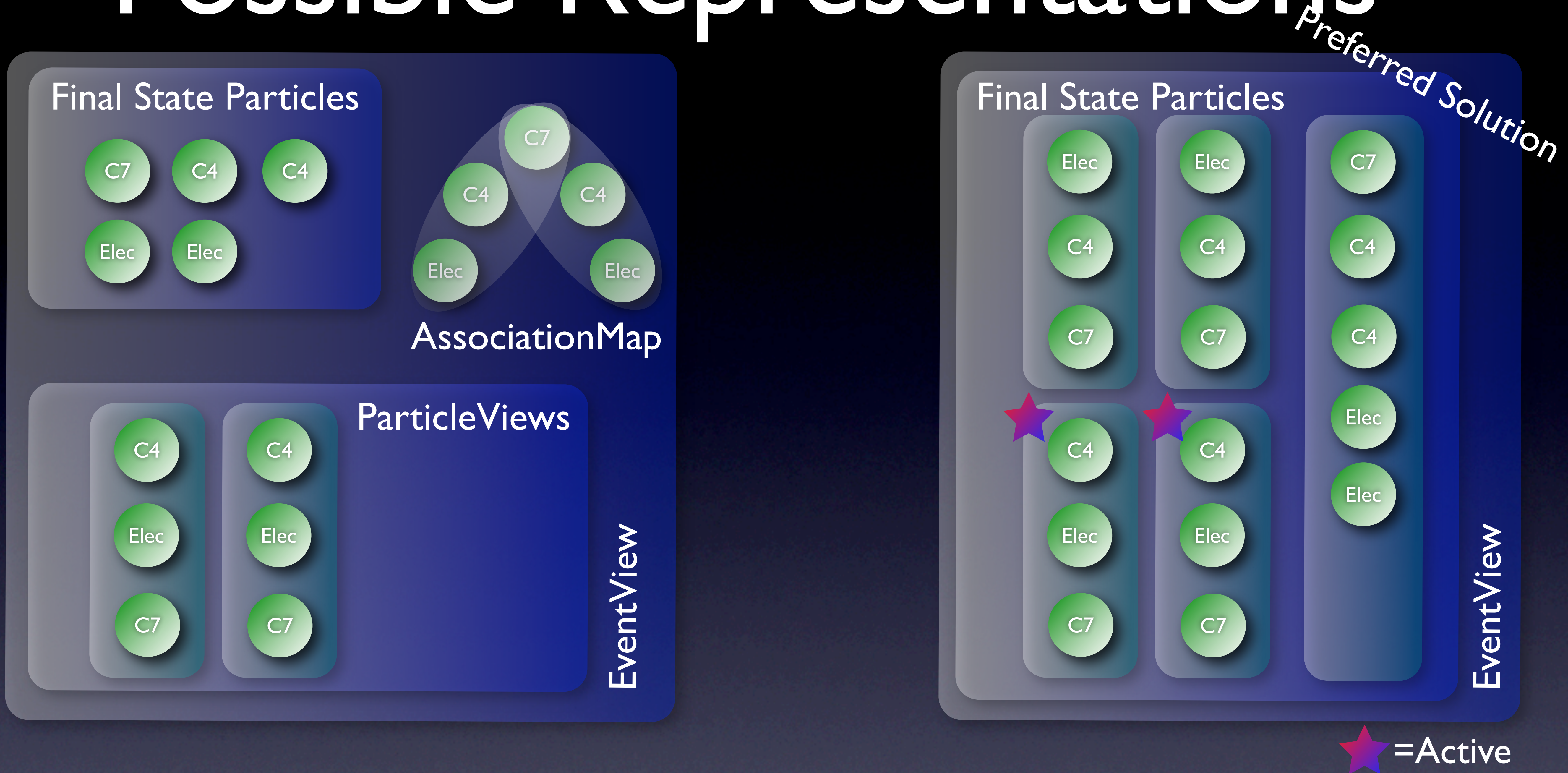


ParticleView/EventView Interplay

- The model which we are going towards:
  - *Object pre-selection*: throw out particles which you definitely will never use.
  - *Overlap determination*: figure out how every object overlaps with other objects
  - *Interpretation*: ask if the event can be interpreted as something (eg 4 jets + lepton, or ttbar, etc ...)
  - *Final selection*: decide if this is a good interpretation of the event.
  - *Re-interpretation*: go from one interpretation to another.



Possible Representations



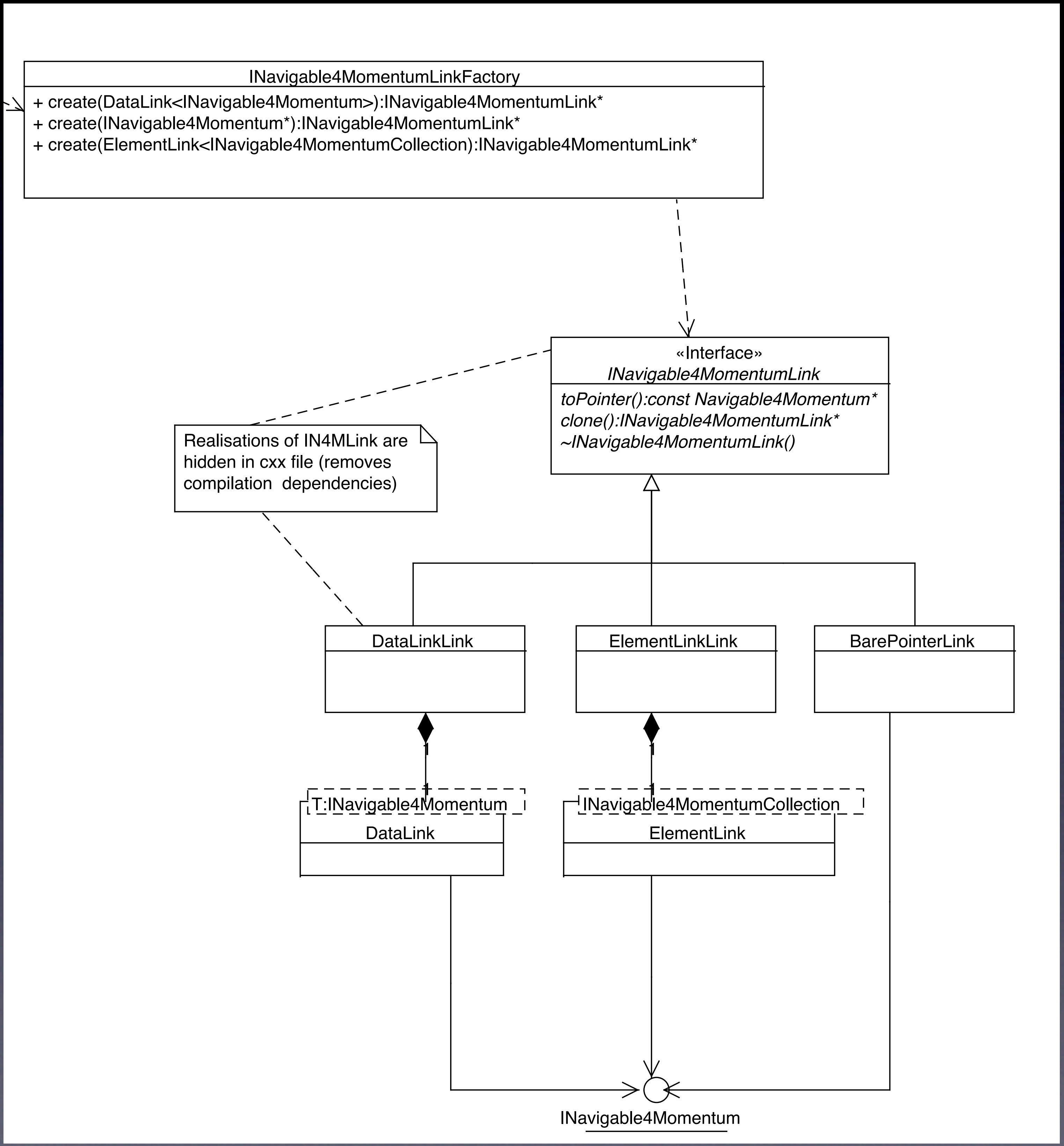
- Add new ParticleView container and AssociationMap object to EventView... save results of preselection & overlap check.
- Tools build PVs at interpretation
- Reinterpretation = get rid of existing PVs

- Build PVs during preselection & overlap check... store in existing FS container.
- Add the ability to mark objects as Active within FS container... interpretation is marking objects active.
- Reinterpretation = mark other PVs active

ILink

- ElementLink is an Athena pointer to an object inside of a container (inside SG).
- DataLink is an Athena pointer to an object inside SG.
- Currently in EV:
  - the Final State Container is a ElementLinkVector because AOD particles are in containers
  - the Inferred Object Container is a vector of DataLinks.
- We would like to point to objects from EventView and ParticleView without worrying about where the objects are stored.
- The original idea for ILink was a base class for ElementLink and DataLink.
- For ParticleView, Peter Sherwood has created a wrapper solution.
- Can be generalized beyond IN4M.

Peter Sherwood



Status Summary

- Full release-12 functionality of EventView is now ready for release-13.
- We can make POOL-based DPDs with EV... today's tutorial.
- We are working on the UserData persistency. This is a very challenging problem.
- Lots of iterations on the design and implementation of ParticleView.
  - New UserDataAccessor and ILink interfaces.

Today's Tutorials

- **You** have a choice:
  - EventViewBuilder tutorial: similar to yesterday... more about how to write/configure tools.
    - Eg: Object Matching
  - Run configure HighPtView/SUSYView: not really a tutorial, but easy instructions to follow and immediately produce custom DPDs.
  - Release 13... building POOL-based DPDs with EventView:
    - SimpleThinningExample
    - HighPtViewDPDThinningTutorial
  - Performance Tutorial: MuonView... example of how to build performance DPD with EventView. (We also have ElectronPhotonView and JetView).

Final Remarks

- We need your support
  - The EventView Framework is still a grass-roots project...
    - no official support from ATLAS... no one is supposed to working on it.
    - no funding!
  - Kyle, Akira, and I developed this despite opposition because we believe in common tools.
    - EventView was our attempt to build a general analysis framework.
    - But this was always meant to be a project for everyone... not something we control.
    - To the surprise of some, we have a very large user base...
    - But maintenance, support, and development is very taxing... and we need to move on.
  - Various people have helped us... but we need ATLAS to take responsibility for the project.
  - Important for you to attend the analysis model forum meeting at the end of this month, and voice your opinion.
    - As in everything, there are good and bad things about EventView.
    - Lets make EventView a better tool for everyone.

Extra Slides

Final Remarks

- There is much more to DPD building than just throwing out info from the AOD.
- Currently, other analysis methods in Athena (eg CBNTAA, AnalysisSkeleton, or SUSYPlot):
  - in many cases must be hardwired for every step of the analysis.
  - provide little structure to allow sharing analyses or standardizing steps.
  - are strongly coupled to the DPD format (ie AANT, POOL-based).
  - usually provide much less native functionality on their DPDs than EV-made DPDs.
  - provide no guarantee that the "UserData" in the DPD (if any) is understandable outside of the group or can be plugged into another analysis.
- The EventView Framework
  - Allows building analyses with as little or as much C++ coding as you like.
  - Provide a huge library of common tools which get tested and validated by user community, and can be easily extend when necessary.
  - Already supports several dozen analyses in all but one physics working group. Supports performance studies too (MuonView, ElectronPhotonView, JetView, ...). And has been running in production (both centralized and private).
  - Will produce "smart" POOL-based DPDs which encapsulate all of the output of analyses in a common format.

Composites

- Composite particles are also a form of UserData.
- With the exception of Top, combinatorics is more an issue for B physics than typical high-p_T analyses.
- But Athena is the natural place for kinematics fits, vertexing, etc...
- And it is impractical to rebuild composites every time you iterate your analysis.
- We need to confirm that CompositeParticle EDM Object is useable in AthenaROOTAccess.
- EventView also stores pointers to CompositeParticles (inferred objects).
- EventViewCombiners allows users to build any decay chain all from job Options:
    - Automatically creates new EventViews if any Composites have any common daughters.
    - Good book-keep mechanism for combinatoric choices.