The Alice AOD analysis model

26 March 2007

Federico Carminati

Statement of the problem

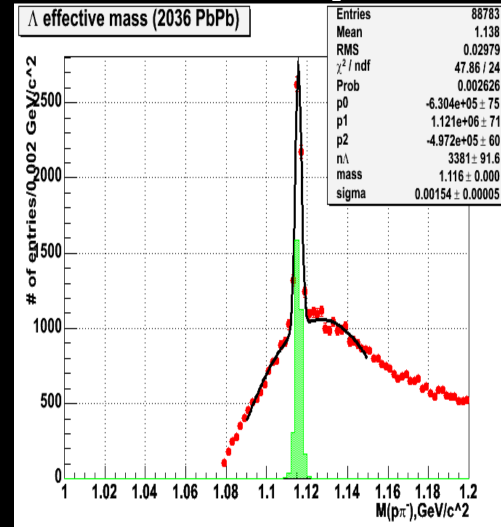
- Analysis is the final aim of all we are doing
- It is also the less tested part of the framework
- Batch analysis has been tested, however it requires AliRoot + AliEn

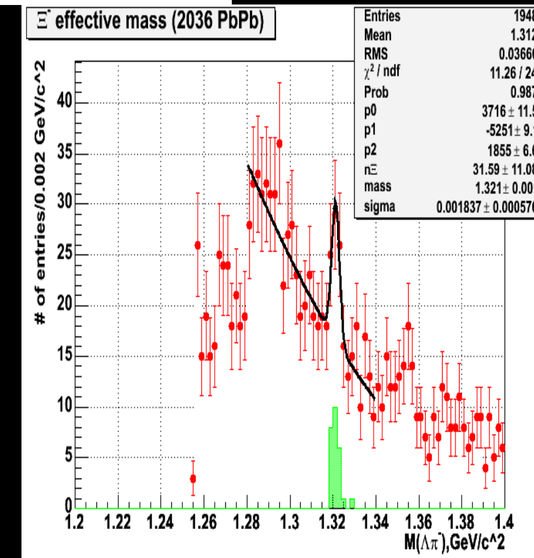
```
df /opt/alien /usr/local/AliRoot/head
178M     /opt/alien
3.4G     /usr/local/AliRoot/NewIO (data files included!)
```

- The “ease” of analysis will help enabling the production of physics results

ALICE Analysis Basic Concepts

- Analysis Models
  - Prompt data processing (calib, align, reco, analysis) @CERN with PROOF
  - Batch Analysis using GRID infrastructure
  - Local analysis
  - Interactive analysis PROOF+GRID
- User Interface
  - Access GRID via AliEn or ROOT UIs
- PROOF/ROOT
  - Enabling technology for CAF
  - GRID API class TAliEn
- Analysis Object Data contain only data needed for a particular analysis
- Analysis à la PAW
  - ROOT + at most a small library
- Work on the distributed infrastructure has been done by the ARDA project





ALICE Data Processing

RAW data

MC data

EMCAL TRD PHOS TOF ZDC HMPID

PMD CRT ITS T0 FMD MUON TPC

Tracks

Clusters

ESD “friends”

ESD

ESD “friends”

AOD(s)

AOD(s) friends

Data reduction in ALICE

	Reco T0/T1s	
RAW 14MB/ev		RAW 1.1MB/ev
Tag 2kB/ev		Tag 2kB/ev
ESD 3MB/ev	Cond Data	ESD 40kB/ev
S-AOD 300kB/ev		S-AOD 5kB/ev

	Analysis T0/T1s/T2/ laptop	
Tag 2kB/ev		Tag 2kB/ev
AODs 300kB/ev		AODs 5kB/ev

Requires AliRoot+Cond+AliEn (once)

Has to run on a disconnected laptop

Tag architecture

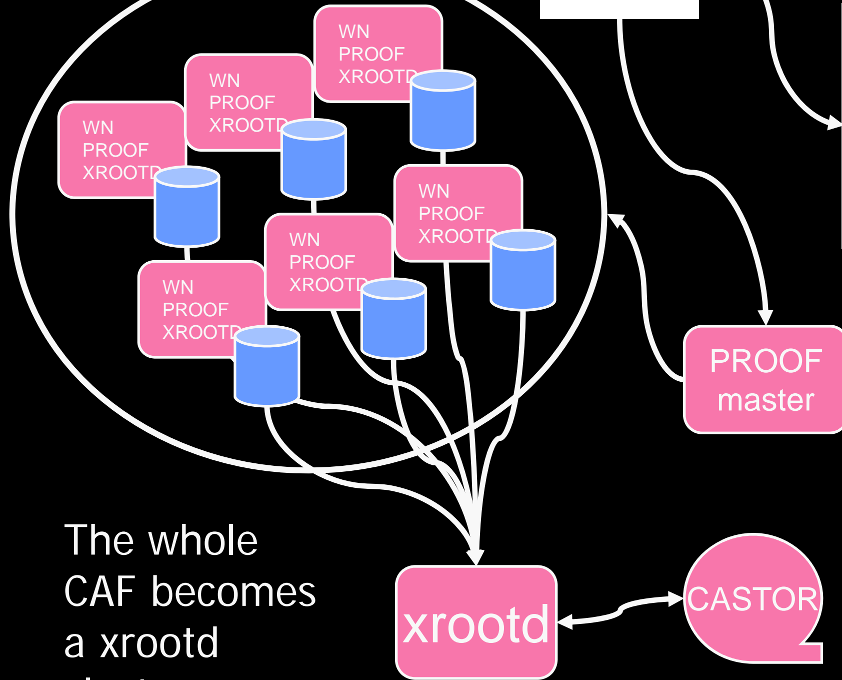
When available

ev#guid	Tag1, tag2, tag3…
ev#guid	Tag1, tag2, tag3…
ev#guid	Tag1, tag2, tag3…
ev#guid	Tag1, tag2, tag3…

BTW, given that this is the ROOT workshop, where are we with this?

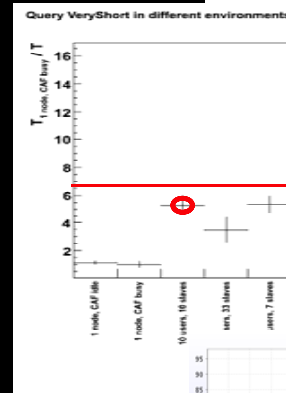
Guid's will mostly refer to AODs

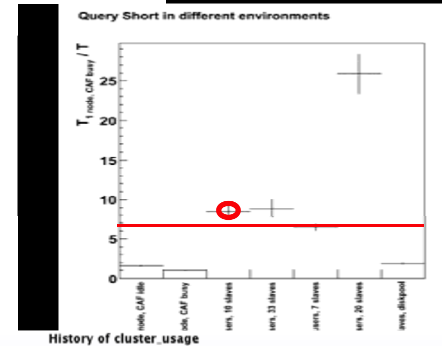
CAF

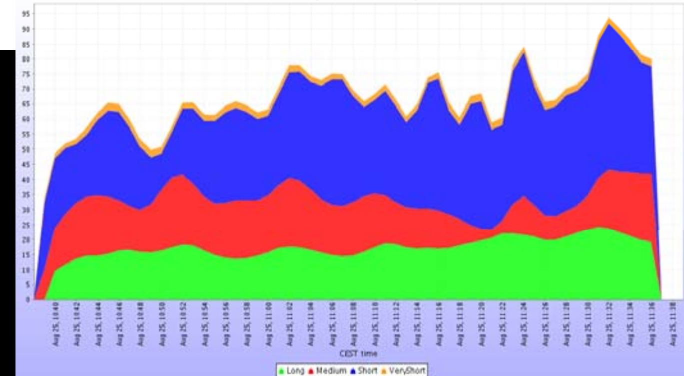


lfn	guid	{se's}
lfn	guid	{se's}
lfn	guid	{se's}
lfn	guid	{se's}
lfn	guid	{se's}

The whole CAF becomes a xrootd cluster







- For offline monitoring and early physics
- Working with the PROOF team on optimisation
- S/W versions handling now in PROOF
- Quotas / load balancing presented here
- Disk space management to be implemented
- ESD and AOD analysis based on TSelector, common base class to be implemented

Once upon a time

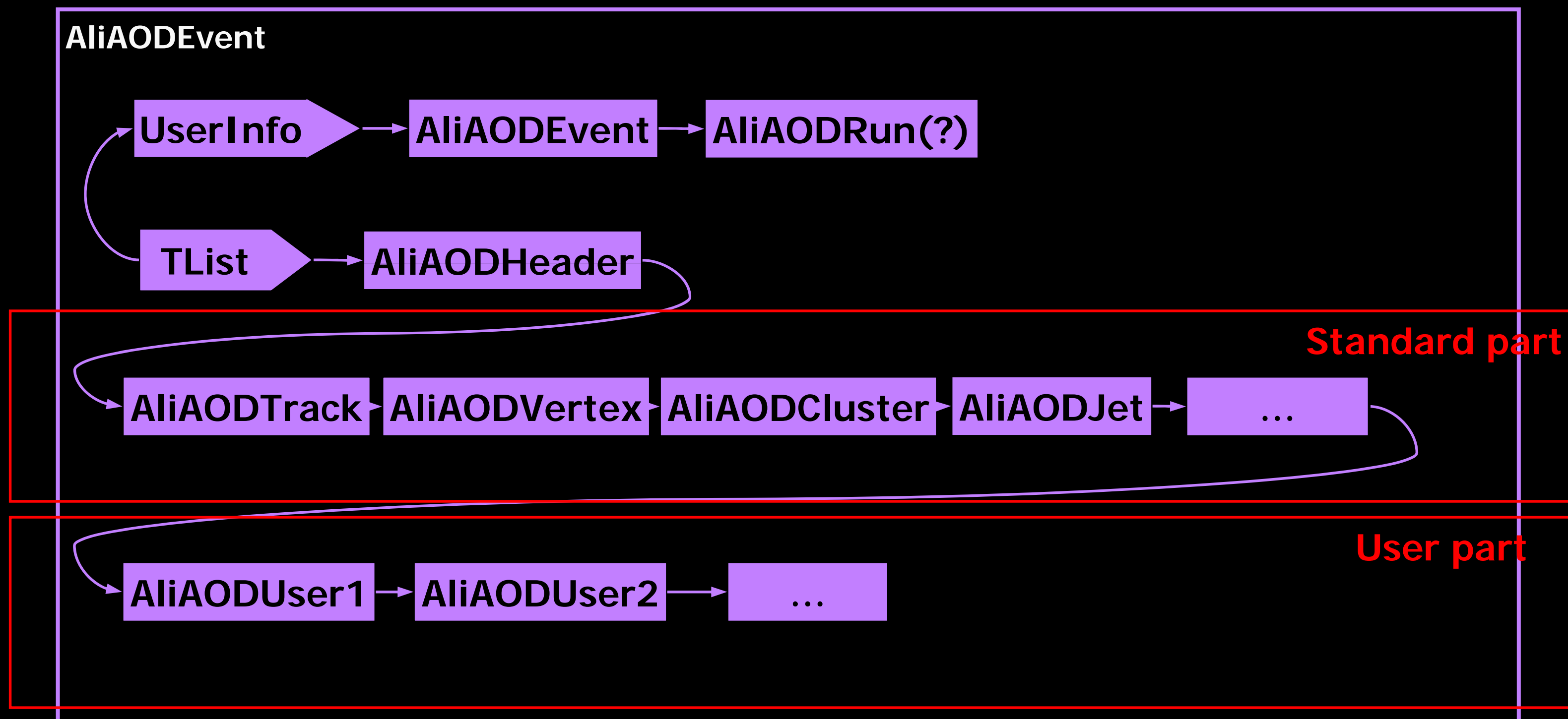
- … there was PAW
- No data model, little or no modularity, difficult to extend
  - … and many other downsides nobody really cared about because it was so user friendly!
- But you would only need a computer with PAW to analyse any data thanks to CWN
- And you could read any version of the data structure – any CWN
- Can we have this with ROOT now?
  - Without losing all the advantages?

Requirements of a design

- Small library independent from AliRoot
  - Easy to be packaged as a par file
- Maximum flexibility for extension customisation and forward/backward compatibility
  - Different working groups will need their own version of AODs
- Keep maximum commonality
- Perform a standard analysis on “any”derived AOD
- Obtain optimal compression on disk
- Possibility to read “only what you need” in memory

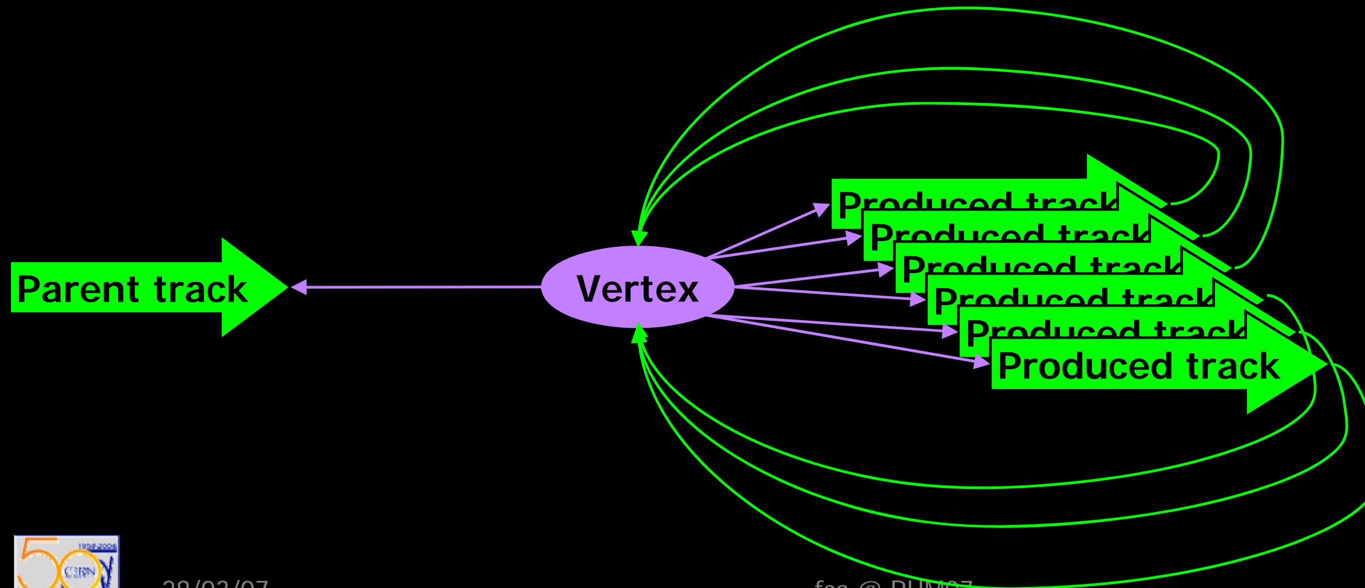
General idea

- Too simple to be true… use a list



Implementation details

- Heavy use of Double32_t types
  - Wouldn't be nice to have also an Int32_t, Short32_t, Float32_t and so on?
- Use of constant-templated classes for covariance matrix
  - CINT needs instantiation, no way to have "header-only" classes if you want I/O
- Current size libAOD.so = 1.6MB



Writing the AOD

- We use the new SetBranch that recursively splits a list

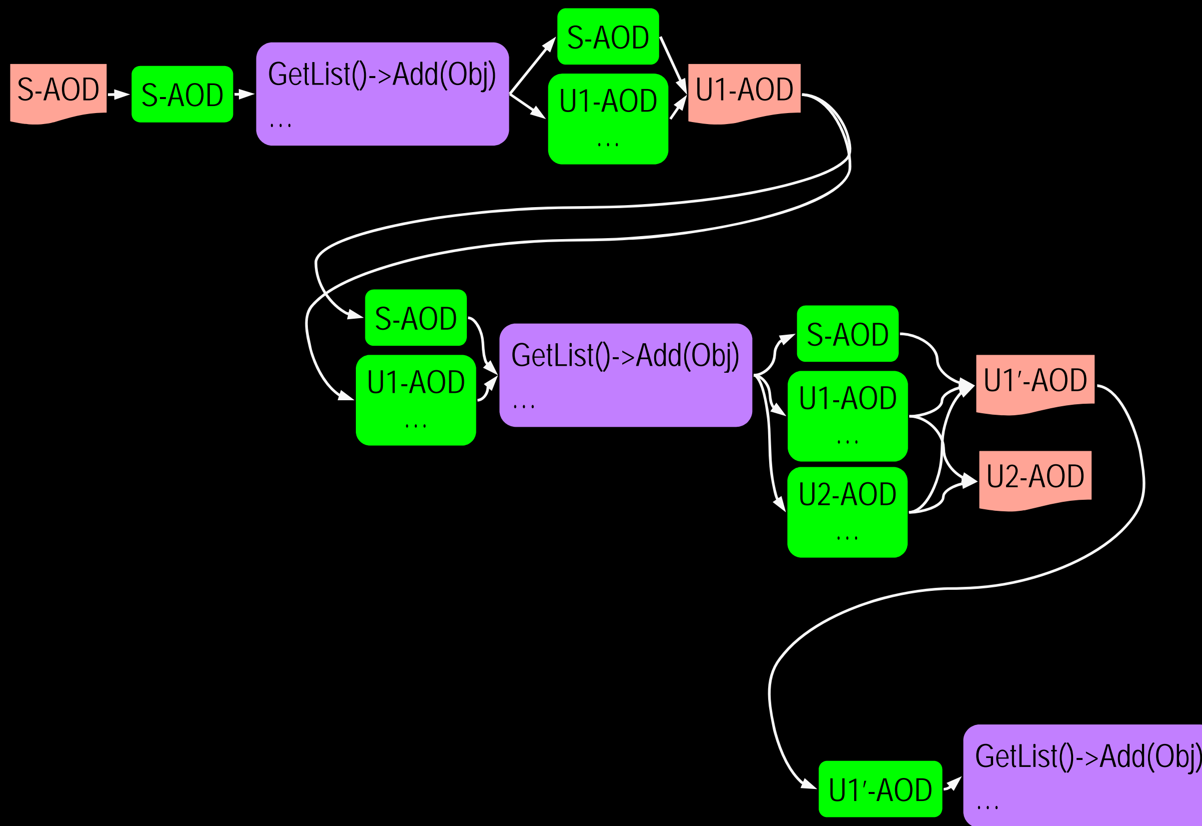
```
// create the tree
TTree *aodTree = new TTree("AOD", "AliAOD tree");
aodTree->Branch(aod->GetList());
```

- So we never write the class itself, only once in the UserInfo
- This is very simple and offers full splitting
  - Experience shows that splitting to level 1 gives best compression

TList branches

```
******************************************************************************
*Tree    :AOD       : AliAOD tree                                            *
*Entries :       10 : Total =          155662 bytes  File  Size =      70361 *
*        :          : Tree compression factor =   2.17                       *
******************************************************************************
*Br    0 :header    :                                                        *
*Entries :       10 : Total  Size=       1628 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=       3200 bytes  Compression=   1.00     *
*............................................................................*
*Br    1 :tracks    :                                                        *
*Entries :       10 : Total  Size=     125758 bytes  File Size  =      58472 *
*Baskets :       10 : Basket Size=       3200 bytes  Compression=   2.14     *
*............................................................................*
*Br    2 :vertices  :                                                        *
*Entries :       10 : Total  Size=       9138 bytes  File Size  =       2885 *
*Baskets :        2 : Basket Size=       3200 bytes  Compression=   2.70     *
*............................................................................*
*Br    3 :clusters  :                                                        *
*Entries :       10 : Total  Size=       1316 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=       3200 bytes  Compression=   1.00     *
*............................................................................*
*Br    4 :jets      :                                                        *
*Entries :       10 : Total  Size=       1216 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=       3200 bytes  Compression=   1.00     *
*............................................................................*
```

Extending the AOD



- You got the idea...

Reading the AOD

- Reading back is “asymmetric”
- We first read the empty event and then we fill the list attached to it

```
AliAODEvent *aod = (AliAODEvent*)aodTree->GetUserInfo()->FindObject("AliAODEvent");
TIter next(aod->GetList());
TObject *el;
while((el=(TNamed*)next()))
  aodTree->SetBranchAddress(el->GetName(),aod->GetList()->GetObjectRef(el));
```

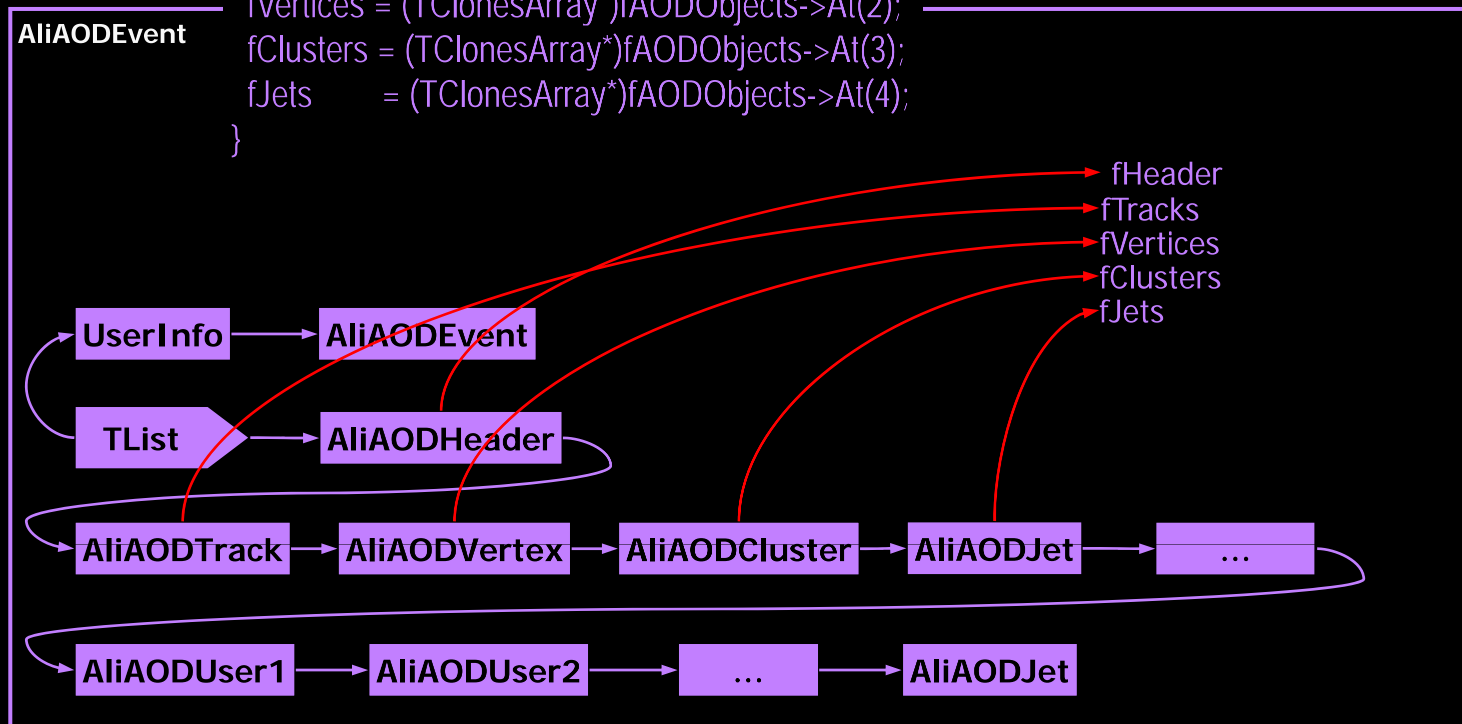
- Although setting branches is a trivial piece of code, it would be nice if this could be automatised
  - At least in case the whole list has to be read
- To avoid to “find” objects in the list every time we access them we have “supporting variables”

Reading back

```
void AliAODEvent::GetStdContent() const
{
  // set pointers for standard content

  fHeader   = (AliAODHeader*)fAODObjects->At(0);
  fTracks   = (TClonesArray*)fAODObjects->At(1);
  fVertices = (TClonesArray*)fAODObjects->At(2);
  fClusters = (TClonesArray*)fAODObjects->At(3);
  fJets      = (TClonesArray*)fAODObjects->At(4);
}
```

```
AliAODTrack* AliAODEvent::GetTrack(Int_t i) const
{ return (AliAODTrack*) fTracks->At(i); }
```



Commonality with ESD / MC truth

- AOD are supposed to be created by ESDs in normal production or in analysis passes
- Same base class with the ESD
  - The same analysis can be done on ESD and AOD for comparison
  - To be implemented :(
- We can generate AODs from ESDs
- We can generate directly AODs from simulated Kine Tree

Better than inheritance?

- Users can add new elements to the list in the same file or in other files
- The existing code will always work also on the derived objects
- Reading can be selective per branch
  - Only needed branches are needed
- Backward compatibility can be enhanced storing source code in the AOD file itself
- Inheritance is not nearly able to capture this richness, as it cannot be defined "on the flight"
- Of course this does not exclude the usage of inheritance for adding new methods or "supporting variables"
  - Shall we "discourage" or "encourage" this?

Any drawback?

- Yes… may be
- You have to know what you are reading…
  - If you do not have the library with the user class in the list you cannot work with it
  - The code could be added to the file, but we did not try this yet operationally
- You have to play with FindObject in the list to get to non-standard variables
- The flexibility of the method can lead to abuse
- Friend files can proliferate and we may have to depend heavily on MetaData in the FC

One example, HF analysis

- Most of heavy-flavour vertexing analyses (D0, D+, Ds+, D*+, Lc, J/y from B, ...) use similar scheme
  - track selection
  - nested loops for pairs, triplets ... building
  - calculate invariant masses, pointing ...
- Common analysis software
  - to produce candidates from tracks (ESD or AOD)
    - AliAnalysisVertexingHF class
  - to store and analyze candidates
    - common base class AliAODRecoDecay
- Main advantages:
  - all candidates produced with one grid production on the data
  - easier to mantain code
  - easier to implement new analysis

AliAODRecoDecay

AliAODVirtualEvent — libAOD

AliAODRecoDecay — libAOD

AliAODRecoDecayHF — libPWG3

AliAODRecoDecayHF2Prong — libPWG3

D^0,J/ψ from B,...

AliAODRecoDecayHF3Prong — libPWG3

D^+,D_s,Λ_c,...

AliAODRecoDstar

Analysis scheme (AOD → AOD)

AliAODHeader

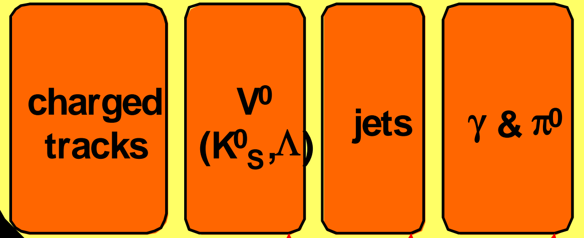
Part of the event Meta Data, e.g.:

- run #, event #, field, trigger, event tags

Global event information, e.g.:

- centrality, multiplicity

Set of TLists:



General AOD

Aod→GetList()→Add(aod2prong)

Aod→GetList()→Add(aod3prong)

Disable branches

AliAODHeader

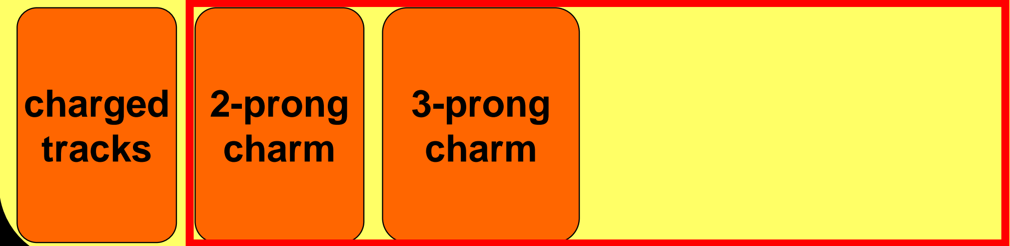
Part of the event Meta Data, e.g.:

- run #, event #, field, trigger, event tags

Global event information, e.g.:

- centrality, multiplicity

Set of TLists:



VertexingHF AOD

MC truth in simulation

- We do not deal with MC truth in code that will run on data
- Use derived classes (à la NA49, STAR)
- This also is made easier by the AOD structure

```
AliAODRecoDecayHF2ProngMC :
public AliAODRecoDecayHF2Prong {
private:
        Int_t *fPDG, *fPDGmother, *fPDGgrandmother, fSignal, AliAODVertex
          *fSecVtxMC ...


AliAnalysisVertexingHFMC :
public AliAnalysisVertexingHF {
AddMCInfo()  // adds PDG codes to the candidates
```

- This could also be handled adding other elements to the list

Conclusion

- A simple AOD schema has been introduced by ALICE
- It goes very close to analysis “à la PAW” without losing ROOT advantages
- One of the first steps will be to create AODs from all the ESDs of the data challenge
- More operational experience is needed to continue the development
- We are planning to recast the ESDs in the same format