Correction FW:
Interaction with the Analysis FW

S.Arcelli, I.Kraus, A.Mastroserio, R.Vernet

- Introduction
- current structure/implementation
- Interaction with the Analysis FW/integration in aliroot

Analysis Session, ALICE Offline Week, 10 Oct 2007

Correction Framework
General Purpose

- Provide general utilities to assist the user in deriving the efficiency corrections (& QA-monitoring) for event-level selections (trigger requirements, global observables, etc..) and/or particle-level corrections (overall efficiency plus intermediate steps as acceptance, reconstruction, effect of user selection criteria…).

- Aim at being able to cover the needs of the majority of the physics analyses, with focus on early ALICE physics and measurements in the central acceptance region. Our "priority" list in developing the framework:
  - single particle analyses
  - V0's (actually, proceeding in parallel with single particles)
  - resonances, decays,…

Correction Framework
Main Requirements

- Possibility to handle N-dimensional grids with user defined dimension and binning (in general, with N>3 sensitive variables on the grid)

  → *"Container" Classes*

  Used to store the MC and the real data on multidimensional containers at different selection levels while looping on events, to manipulate this information to derive the correction maps, and to get the corrected data

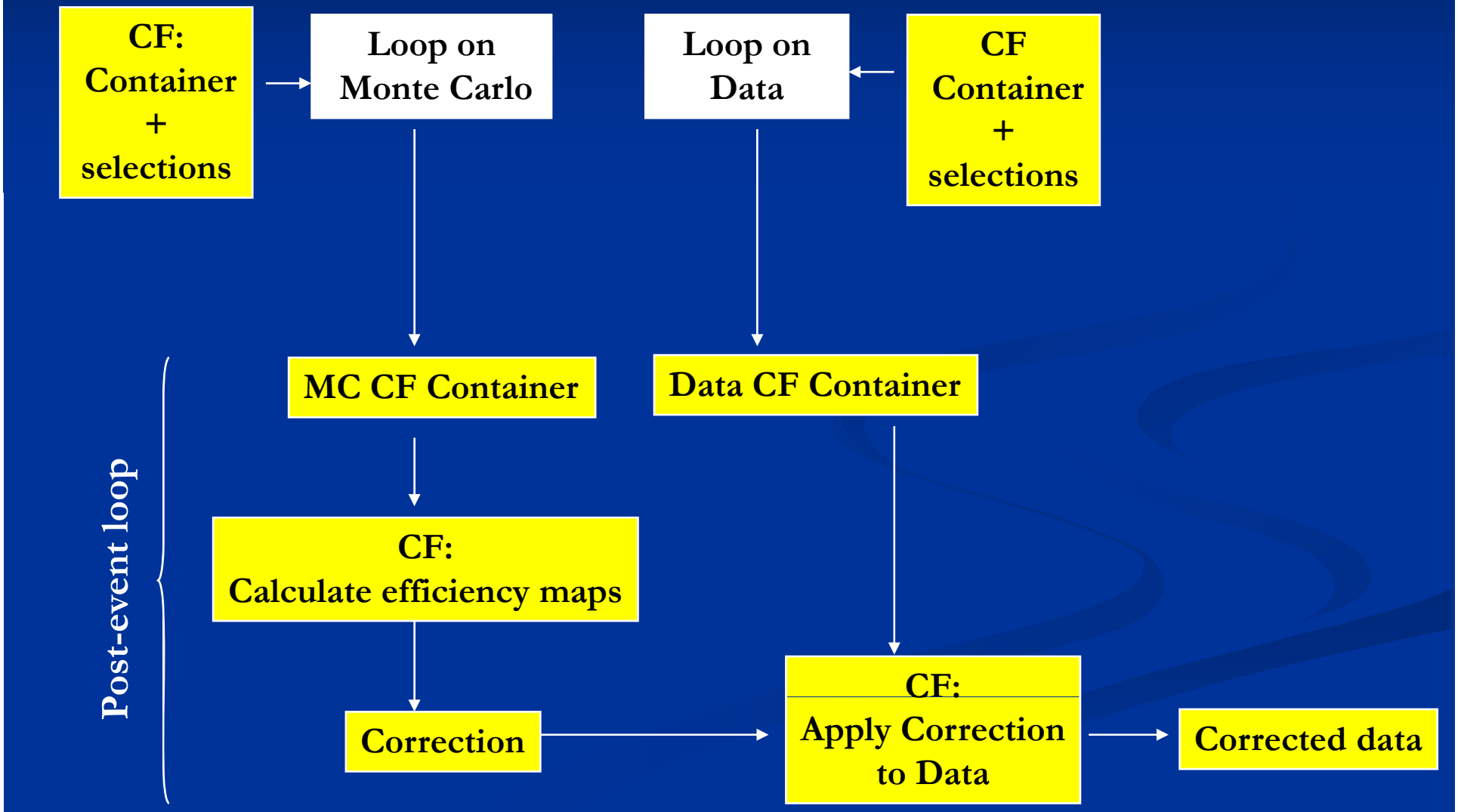
- Interface for "general use-case" selections. For example:
  - Impose generator-level requirements (for ex., ask for true primary tracks)
  - Check if a particle (stable or decayed) is in the acceptance
  - Reconstruction and PID efficiencies
  - Trigger efficiency for different event classes/trigger selections

  At the same time, let the user the freedom to introduce his own cuts in a transparent way via the same interface.

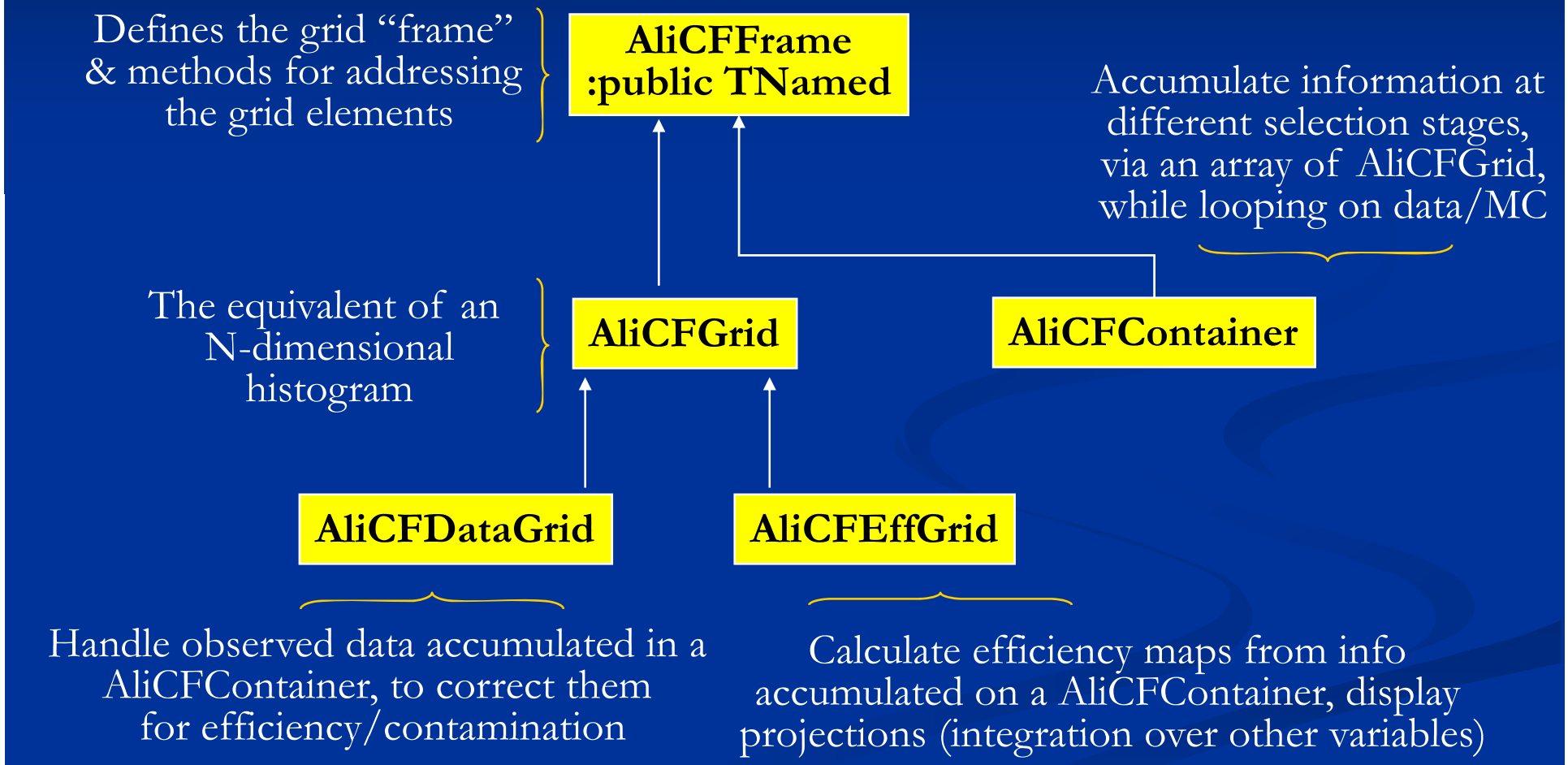
  → *"Selection" Classes*

Correction Framework

General flow of the correction process



The "Container" Classes



The "Container" Classes

- To store the data, the N-dim grids as implemented in AliCFGrid use arrays of floats of dimension $\prod_{i=1,Nvar} N_{bins}(i)$. For large number of bins and dimensions, the size in memory becomes large, independently on the occupancy of the cells. Concerns arise in particular in the context of organized analysis, with several tasks handling large correction maps

- Started to discuss this issue with ROOT team in mid-July . A prototype for a class to handle N-dim grids has been committed to the ROOT repository the last week (A. Neumann)

The “Container” Classes

- THnSparse: (http://root.cern.ch/root/html/THnSparse.html): memory is allocated only for the cells that have non-zero content. Optimal in case one needs to handle a large number of cells, and the occupancy of a significant fraction of cells is very low.

- Still missing functionalities, like correct error handling in projections, operations (Add, Multiply, Divide...), rebinning and interpolation (these last two items also missing in AliCFGrid).

- Plan to make some additional tests on performance (vs AliCFGrid), cooperate in developing it, and eventually replace the current AliCFGrid implementation. Should be rather transparent for the rest of the Container Classes

The "Selection" Classes

A number of selections classes have been prepared, also taking as a reference a number of selections already implemented in the PWG0 repository:

- Particle-level selections
  - Generator-level selections. Kinematic quantities, being a primary
  - Acceptance level selections (access to track references via AliMCEventHandler)
  - Reconstructed level cuts on AliESDtracks, AliESDV0's. Kinematic variables, quality cuts, conditions on being a primary, dedicated PID class
- Event Level selections:
  - At the generator level (vertex, MB MC process type)
  - At the recostructed level (vertex, trigger)

Via the selection classes, we also foresee to accumulate diagnostic histograms for correction QA on the cut variables (as for example done in AliESDtrackCuts)

The "Selection" Classes

Implemented in the form of elementary cut classes, each category inheriting its own base class (combination of cuts is handled as an TObjarray of elementary cuts:

- for generator/acceptance particle level cuts:
  - 1 member function virtual Bool_t Pass(TParticle*) , used to apply the cut
  - 1 member function virtual Bool_t Pass(Int_t index), to select on track references
  - 1 member datum pointing to the current AliMCEventHandler* ,to navigate over the stack
- for reconstructed particle level cuts:
  - applied on AliESDtrack's
    - 1 member function virtual Bool_t Pass(AliESDtrack*)
    - 1 member datum pointing to the current AliESDEvent*
  - Same structure as above for V0's
    - 1 member function virtual Bool_t Pass(AliESDv0*)
    - 1 member datum pointing to the current AliESDEvent*
- for event-level selections:
  - Generator level
    - 1 member function virtual Bool_t Pass(AliGenEventHeader*)
  - Reconstructed level
    - 1 member function virtual Bool_t Pass(AliESDEvent*)

The User Job…

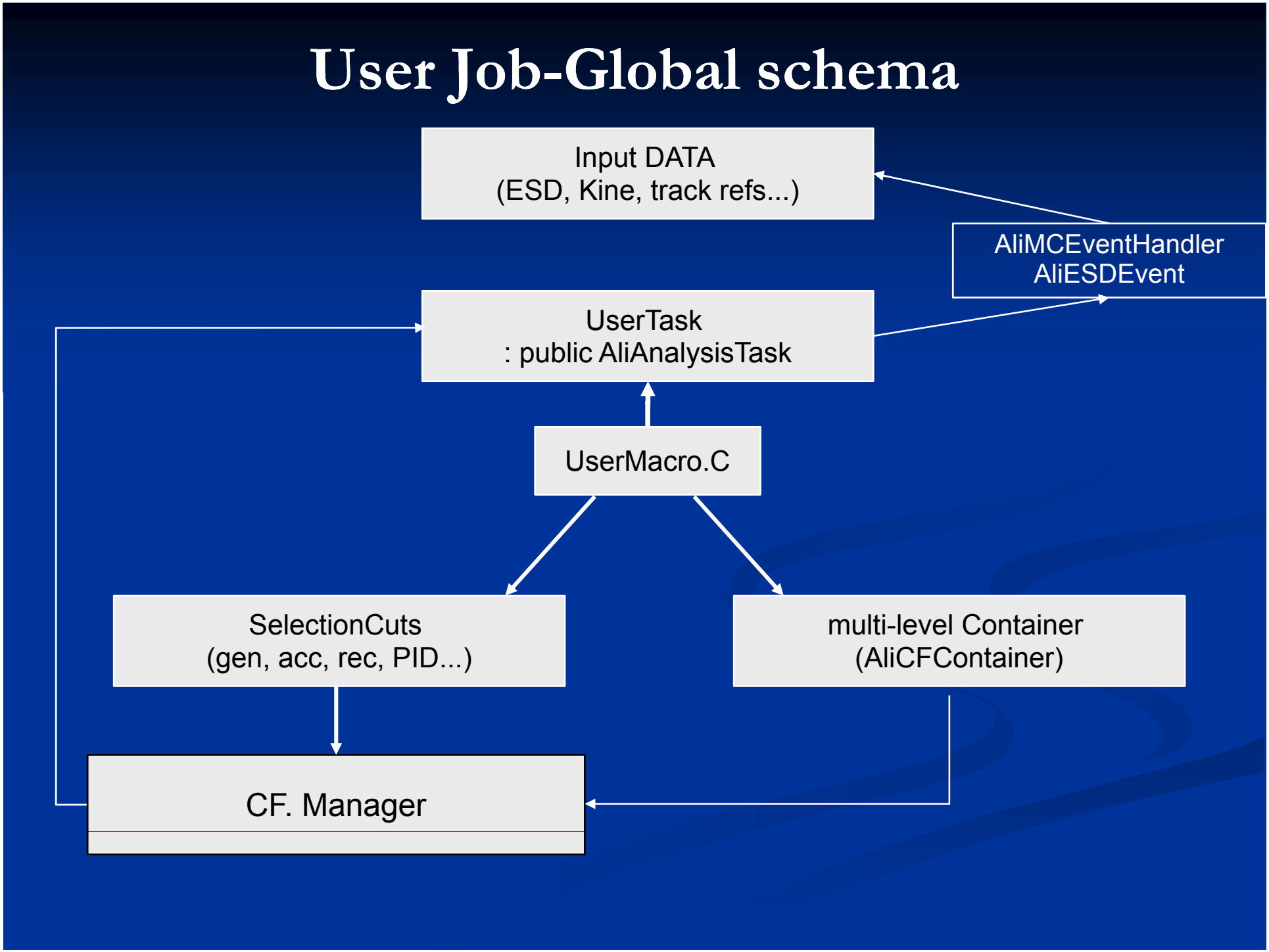
- The user has to setup 2 things
  - A Task (deriving from) AliAnalysisTask
  - a initialisation macro
- Macro setup according to :
  - his container
    - number of selection steps (generated, in acceptance, reconstructed, selected...)
    - number of sensitive variables for acceptance/efficiency calculation (pT, y...)
  - the data he wants to loop on
  - the cuts he wants to apply on his tracks
    - PID, momentum, number of track references...
  - the task to use
- Task implementation :
  - user writes what he wants to do to calculate his efficiency
    - pass lists of cuts
    - fill the container

The User Macro…

- The list of cuts to be applied are defined in the user macro
- Several steps in the efficiency calculation -> several kinds of cuts
  - generation , acceptance, reconstruction, selection (PID, for ex)
- AliCFManager: interface class used to pass the container & the list of selections to the task, providing methods to check the cuts (or a subset of).

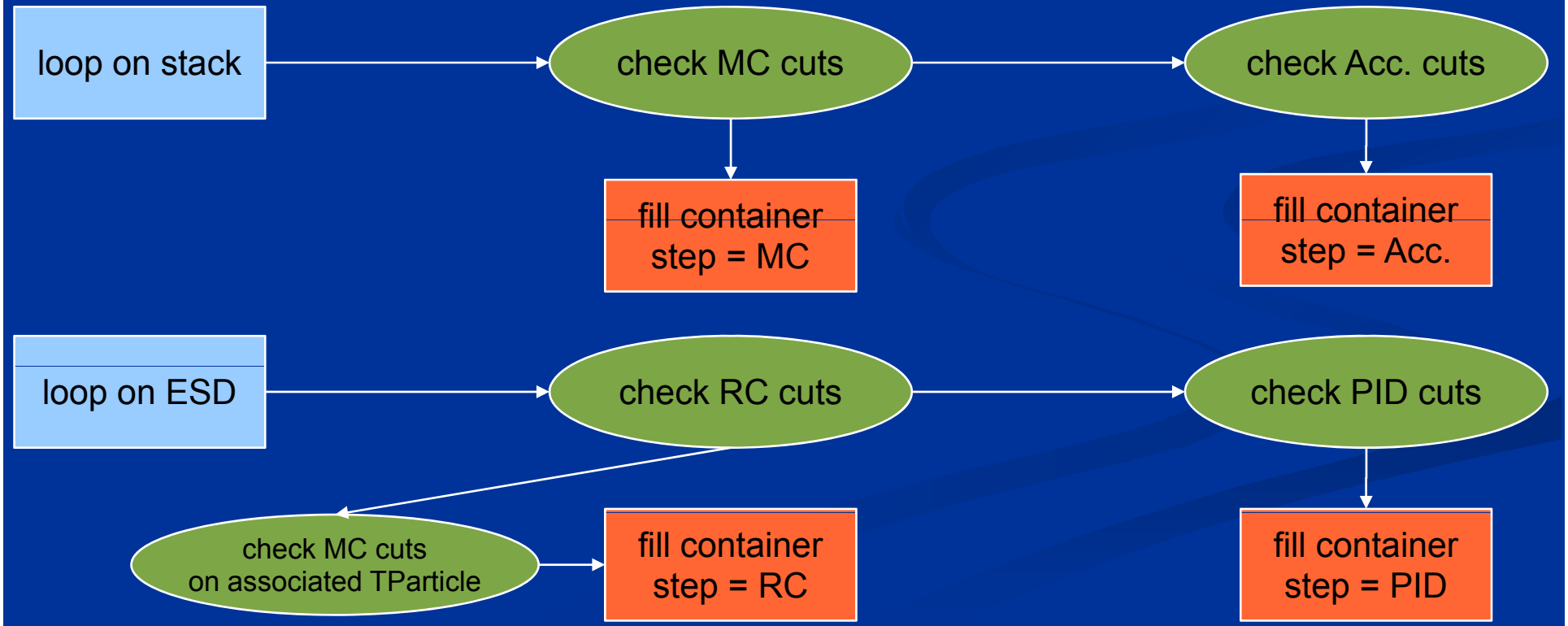
```
//------ sets the container and lists of cuts to the Correction Framework manager -------
AliCFManager* IO = new AliCFManager() ;
IO->SetContainer     ( container       );          //container
IO->SetEvtGenCutList ( SetupEventGenCuts     ());    //kind of MC process
IO->SetEvtTrigCutList( SetupEventTriggerCuts ());    //trigger level
IO->SetEvtRecCutList ( SetupEventRecCuts     ());    //prim. vertex...
IO->SetPartGenCutList( SetupGenCuts          ());    //cuts on generated particles
IO->SetPartAccCutList( SetupAccCuts          ());    //cuts on track references
IO->SetPartRecCutList( SetupRecCuts          ());    //cuts on reconstr. tracks (ESD)
IO->SetPartRecCutList( SetupSelCuts          ());    //cuts on selected. tracks (ESD)


TClonesArray* SetupGenCuts() { //setup generation-level cuts
  AliCFParticleCutPt*     cutPt     = new AliCFParticleCutPt("cutPt","MC_PT") ;
  AliCFParticleCutCharge* cutCharge = new AliCFParticleCutCharge("cutCharge","MC_CHARGE");
  cutMCPt->SetPtRange(0.0,10.0) ; cutCharge->SetCharge(-1.);
  TObjArray* fCutList = new TObjArray(0) ;
  fCutList->AddLast(cutMCPt);
  fCutList->AddLast(cutCharge);
  return fCutList ;
}
```

User Job-Global schema



User Task…

- derives from AliAnalysisTask
- use of AliMCEventHandler
  - easy access to kinematics and track references
- the Task does :
  - loop on events
  - select Events and particles/tracks using previously defined cuts
  - fills the container and save it in an output file



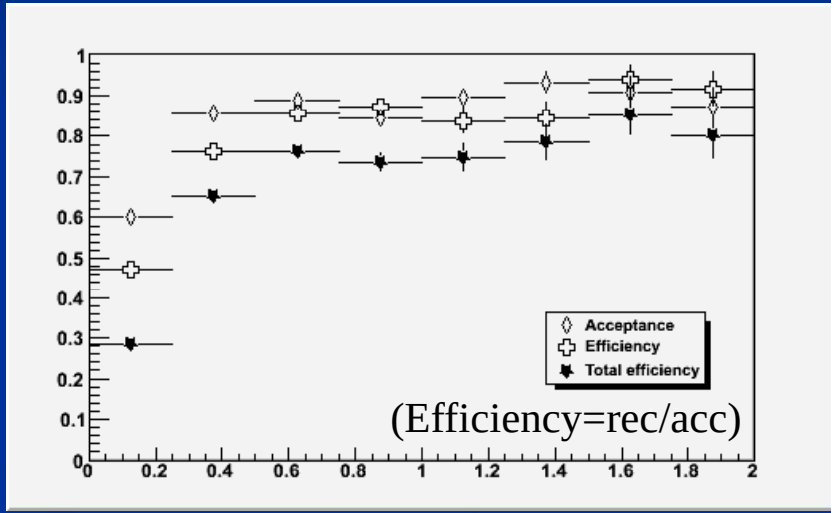
User Task…

Inside the event loop:

```
void AliSingleTrackTask::Exec(Option_t *) {
 ..... // get mc event handler (mcTruth)
 AliStack* stack = mcTruth->Stack();
 for (loop_on_stack) {
   TParticle* particle = ... ;
   if (!fCFManager->CheckPartGenCuts(part,"all")) continue;   // selection cut Generation
            ...// fill container for step “Generated”
   if (!fCFManager->CheckPartAccCuts(index,"all")) continue ; // selection cut Acceptance
            ...// fill container for step “in Acceptance”
 }
 for (loop_on_ESDtracks) {
   AliESDtrack* track = ... ;
   TParticle* part=stack->Particle(track->GetLabel());       //get the associated MC
   if (!fCFManager->CheckPartRecCuts(track,"all")) continue; // selection cut Reconstructed
    if (fCFManager->CheckPartGenCuts(part,"all")){
      // check if mc passes the cuts
      // …fill container for step “Reconstructed”}
   if (!fCFManager->CheckPartSelCuts(track,"all")) continue;  // selection cut on PID
   if (fCFManager->CheckPartGenCuts(part,"all")){
     // check if mc passes the cuts
     // …fill container for step “Selected”...}
 }}
```

Single Track Analysis
first tests on AliEn

- An example of single track analysis Task has been tested on AliEn, for a very first validation
- Data sample:
  - Pythia pp @14 TeV
  - 1000 events
- gen. selection of primary π^- in $|y|<1$
- acceptance defined by #track.refs.
  - >= 3 in ITS
  - >= 1 in TPC
- ESD track must have #clusters in TPC > 50



→ the general schema is working
Needs more appropriate reference selections
in acceptance calculation

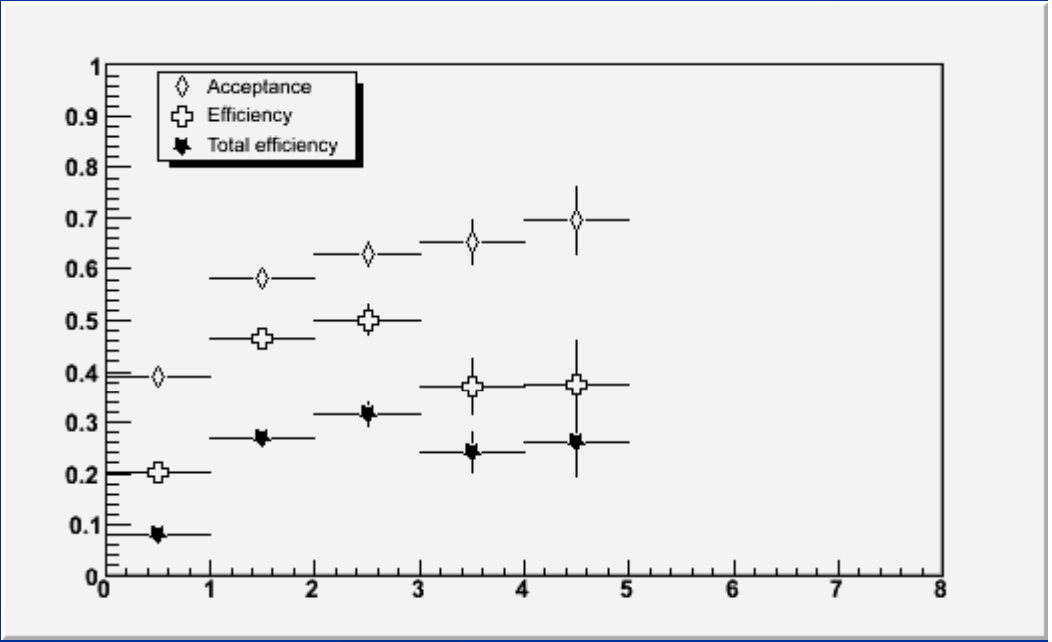
V^0 Analysis

first test on AliEn

- 24 K event sample, p-p MB @14 TeV
- V^0 MC selection: comes from a K^0_s in $|y|<1$
- V0 acceptance selection: both daughters with >0 TPC track refs
- V^0 ESD test selection :
  - pt=0 → 5 GeV/c
  - $|y| < 1$
  - dca < 1 cm
  - b+,b- > 0.1 cm
  - cos(P.A.) > 0.995

→ V0 code is working

still several updates and improvements to be done



First Feedback

The current implementation has recently been discussed with A.Morsch And F.Carminati, who provided us a number of suggestions, mainly related to the selection classes design (many thanks for their feedback):

- Using elementary cut classes provides the highest flexibility to the user, but has a disadvantage in terms of code organization and readability.
  - Cluster the cuts according to the type of 'physical' selection : quality cuts, cuts identifying a particle as primary, basic kinematic cuts,….
  - For what concerns kinematic cuts, in view of foreseen common inheritance one could actually really converge on a general unique implementation applicable to AliMCParticle,AliESD(AOD)track, AliESD(AOD)v0's, AliAODRecoDecay

- Most of the selections which have been implemented can derive from the AliAnalysisCuts base class in ANALYSIS:
  - 1 member function virtual Bool_t IsSelected(TObject*)

→ **Some changes are straightforward, other are part of an evolving picture, and will have to proceed in steps...**

Correction Framework
Summary/Plans

A very first version of the basic components of the CF (containers, selections, a prototype for an interface class) has been developed

- In view of a future integration in aliroot, some of them require a redesign:
  - Inheritance from AliAnalysisCuts in selection classes
  - Group sets of elementary selections classes according to cathegories (rec. quality, physics selection, purely kinematic cuts..)
- Some remarks/questions:
  - Selection on track reference information cannot be handled by AliAnalysisCuts, for the moment, due to the type of argument which is passed. Keep it separate? Introduce a IsSelected(Int_t index) in AliAnalysisCuts?
  - Selection classes acting on multiple objects (pair of tracks, for ex, in resonance analysis), how are we going to treat them?

Correction Framework
Summary/Plans

- More on Acceptance Calculation: further investigate if the existing track reference information is appropriate for a proper definition of "findable track" standards (PPR type) in each system (for example, in the case of TPC one needs further 'manipulation') .

- Once this is sorted, may derive some 'higher-level' info to be stored in AliMCParticle (to be defined)? Will be a ready-to-use information for acceptance calculation (but probably will not cover all the needs of an user analysis?)

- Apart from changes in the implementation of selection classes, we still need to test the correction schema running it also on the CAF (only AliEn so far)