ALICE Offline Tutorial

Alice Core Offline

3 March 2010

Part II
Analysis framework

Andrei Gheata
Christian Klein-Bösing
Andreas Morsch

These slides + examples:
http://aliceinfo.cern.ch/Offline under “Documentation”

Prerequisites

- Copy the tutorial tarball locally:
  - > wget http://morsch.home.cern.ch/morsch/analysis-tutorial.tgz

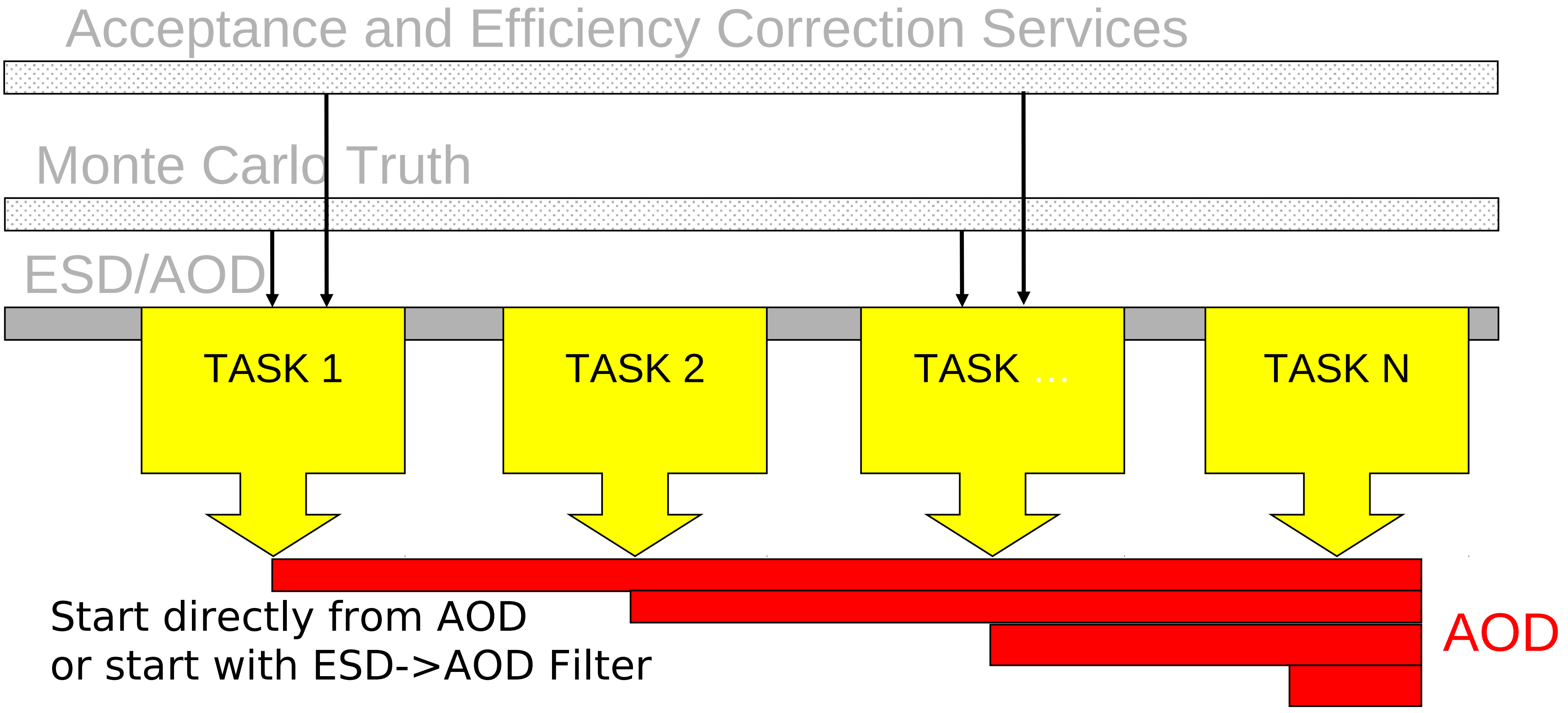
Analysis

- Software
  - AliRoot
    - Specialized ROOT for ALICE
    - AliRoot = ROOT + ALICE libraries
  - Your code
- What is the data
  - Usually ESD, AOD or Monte Carlo kinematics (MC truth)
- Where does your analysis code run?
  - Local = On your machine
  - In PROOF ("Parallel ROOT Facility")
    - Parallel analysis on a cluster
    - Not related to the Grid
  - In the AliEn ("Alice Environment") Grid
    - AliEn is the software of ALICE to access the Grid
    - As a user job or in an **organized analysis**

What is Organized analysis?

- Centrally coordinated analysis “train”
  - Collected analysis tasks (“train-wagons”) pass over the data
  - No chaotic request of data
- Most efficient way for many analysis tasks to read and process the full data set.
  - In particular if resources are sparse.
  - Optimise CPU/IO ratio
- User can rely on previous “service” and PWG tasks
  - Check for data integrity
  - Filters (Eg. ESD->AOD)
  - One task to find e.g. Jets
- But also…
  - Helps to develop a common well tested framework for analysis.
  - Develops common knowledge base and terminology.
  - Helps documenting the analysis procedure and makes results reproducible.

The plan: Analysis train producing AODs



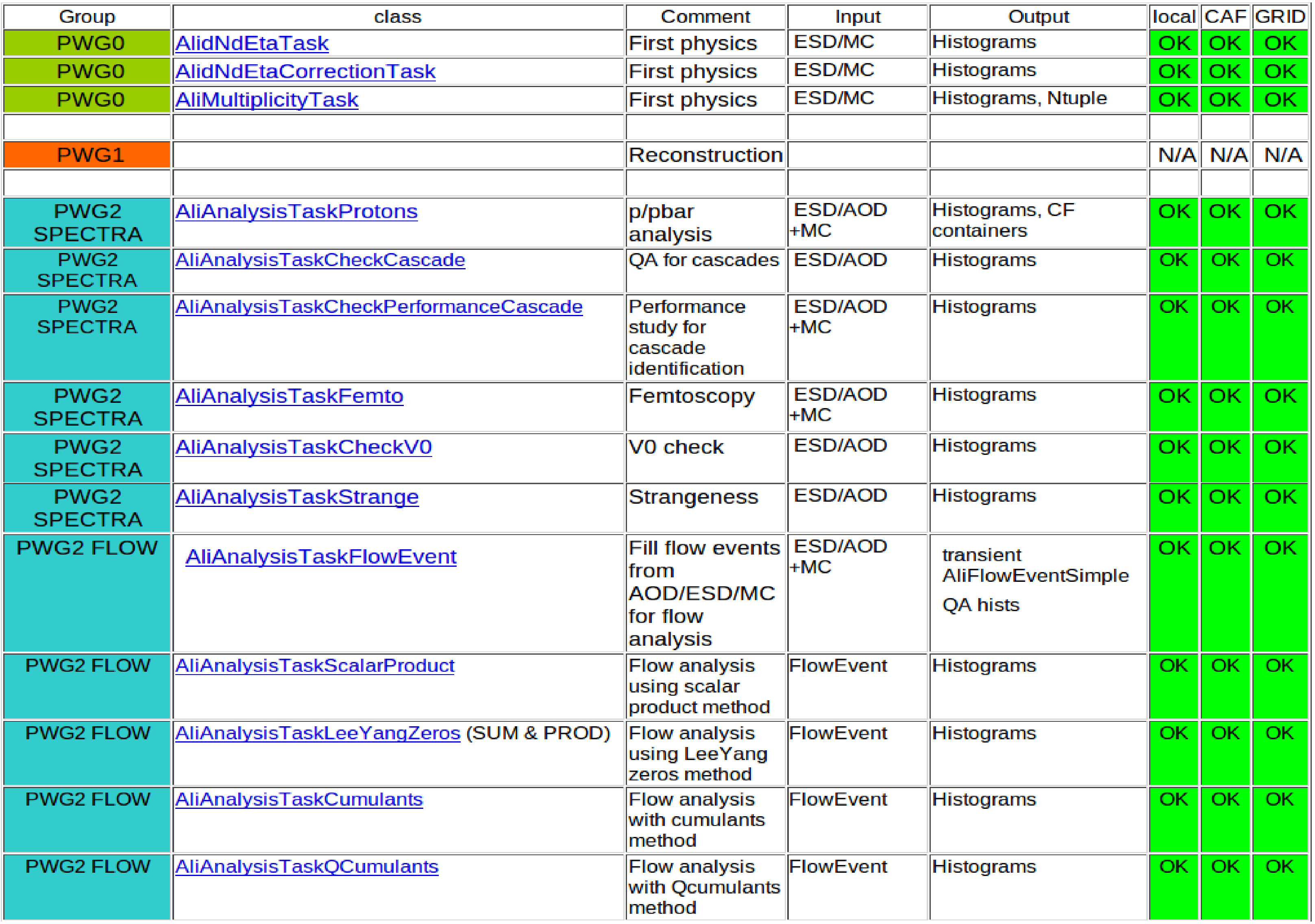
Start directly from AOD
or start with ESD->AOD Filter

Possibility to write DeltaAODs

Information and help

- Train status
  - http://pcalimonitor.cern.ch/train_details.jsp
- Savannah
  - https://savannah.cern.ch/projects/pdc/
- Analysis Mailing List
  - alice-project-analysis-task-force@cern.ch
- Documentation
  - http://aliceinfo.cern.ch/Offline/Activities/Analysis/
- Analysis News
  - http://aliceinfo.cern.ch/Offline/Activities/Analysis/NewsAndProblems.html

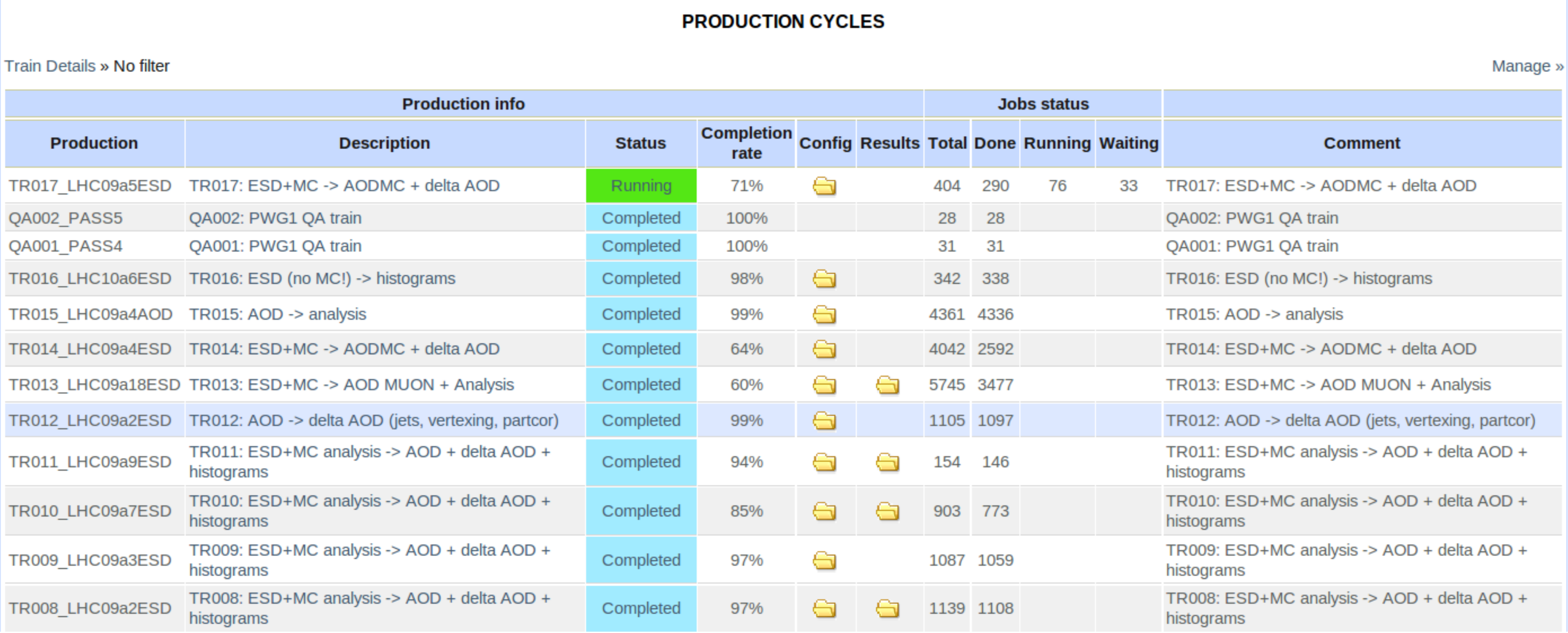
Example: Some train wagons



Group	class	Comment	Input	Output	local	CAF	GRID
PWG0	AlidNdEtaTask	First physics	ESD/MC	Histograms	OK	OK	OK
PWG0	AlidNdEtaCorrectionTask	First physics	ESD/MC	Histograms	OK	OK	OK
PWG0	AliMultiplicityTask	First physics	ESD/MC	Histograms, Ntuple	OK	OK	OK
PWG1		Reconstruction			N/A	N/A	N/A
PWG2 SPECTRA	AliAnalysisTaskProtons	p/pbar analysis	ESD/AOD +MC	Histograms, CF containers	OK	OK	OK
PWG2 SPECTRA	AliAnalysisTaskCheckCascade	QA for cascades	ESD/AOD	Histograms	OK	OK	OK
PWG2 SPECTRA	AliAnalysisTaskCheckPerformanceCascade	Performance study for cascade identification	ESD/AOD +MC	Histograms	OK	OK	OK
PWG2 SPECTRA	AliAnalysisTaskFemto	Femtoscopy	ESD/AOD +MC	Histograms	OK	OK	OK
PWG2 SPECTRA	AliAnalysisTaskCheckV0	V0 check	ESD/AOD	Histograms	OK	OK	OK
PWG2 SPECTRA	AliAnalysisTaskStrange	Strangeness	ESD/AOD	Histograms	OK	OK	OK
PWG2 FLOW	AliAnalysisTaskFlowEvent	Fill flow events from AOD/ESD/MC for flow analysis	ESD/AOD +MC	transient AliFlowEventSimple QA hists	OK	OK	OK
PWG2 FLOW	AliAnalysisTaskScalarProduct	Flow analysis using scalar product method	FlowEvent	Histograms	OK	OK	OK
PWG2 FLOW	AliAnalysisTaskLeeYangZeros (SUM & PROD)	Flow analysis using LeeYang zeros method	FlowEvent	Histograms	OK	OK	OK
PWG2 FLOW	AliAnalysisTaskCumulants	Flow analysis with cumulants method	FlowEvent	Histograms	OK	OK	OK
PWG2 FLOW	AliAnalysisTaskQCumulants	Flow analysis with Qcumulants method	FlowEvent	Histograms	OK	OK	OK

http://aliceinfo.cern.ch/Offline/Activities/Analysis/AnalysisFramework/index.html

Analysis Train on MonaLisa



PRODUCTION CYCLES

Train Details » No filter

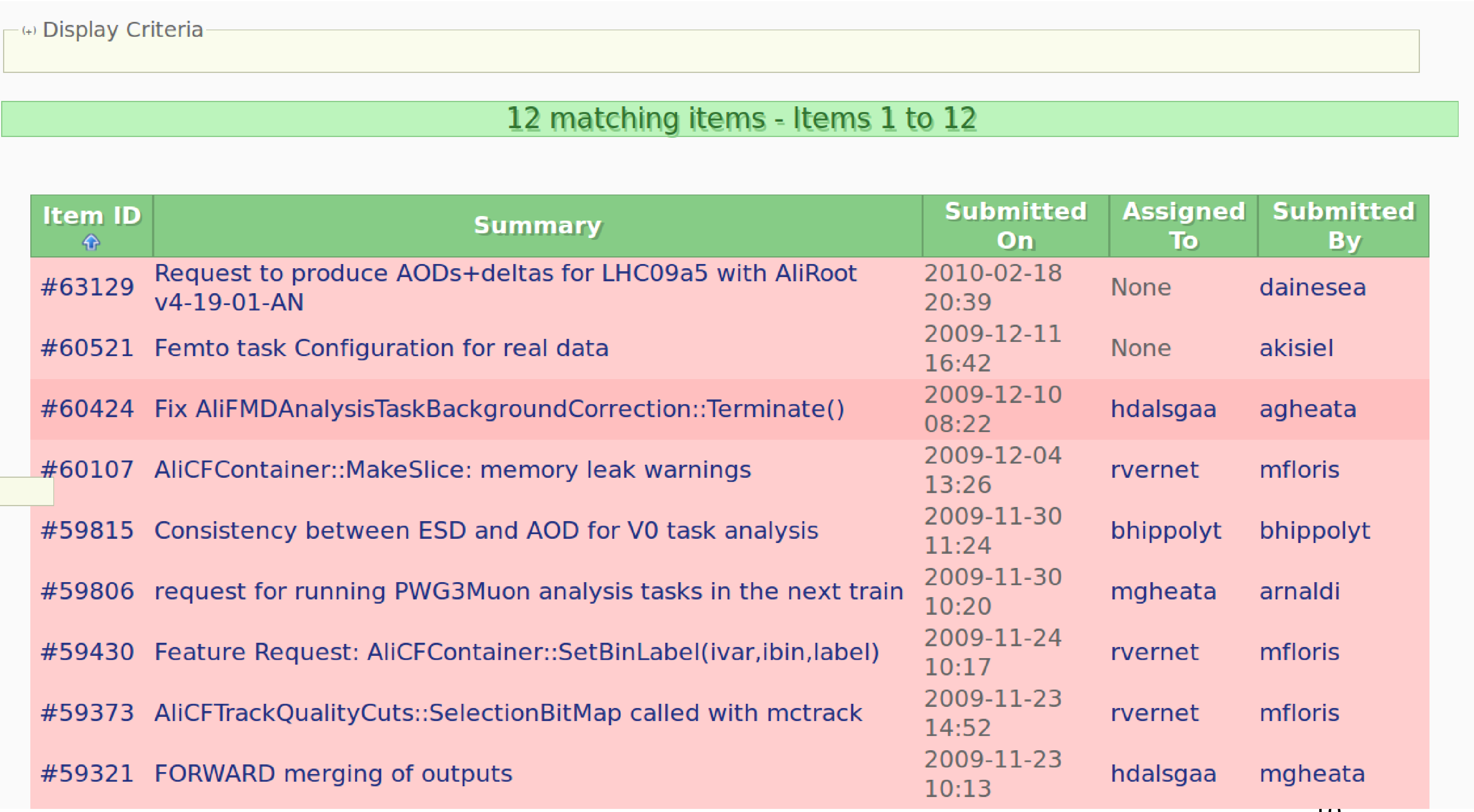
Manage »

Production info						Jobs status				
Production	**Description**	**Status**	**Completion rate**	**Config**	**Results**	**Total**	**Done**	**Running**	**Waiting**	**Comment**
TR017_LHC09a5ESD	TR017: ESD+MC -> AODMC + delta AOD	Running	71%	📁		404	290	76	33	TR017: ESD+MC -> AODMC + delta AOD
QA002_PASS5	QA002: PWG1 QA train	Completed	100%			28	28			QA002: PWG1 QA train
QA001_PASS4	QA001: PWG1 QA train	Completed	100%			31	31			QA001: PWG1 QA train
TR016_LHC10a6ESD	TR016: ESD (no MC!) -> histograms	Completed	98%	📁		342	338			TR016: ESD (no MC!) -> histograms
TR015_LHC09a4AOD	TR015: AOD -> analysis	Completed	99%	📁		4361	4336			TR015: AOD -> analysis
TR014_LHC09a4ESD	TR014: ESD+MC -> AODMC + delta AOD	Completed	64%	📁		4042	2592			TR014: ESD+MC -> AODMC + delta AOD
TR013_LHC09a18ESD	TR013: ESD+MC -> AOD MUON + Analysis	Completed	60%	📁	📁	5745	3477			TR013: ESD+MC -> AOD MUON + Analysis
TR012_LHC09a2ESD	TR012: AOD -> delta AOD (jets, vertexing, partcor)	Completed	99%	📁		1105	1097			TR012: AOD -> delta AOD (jets, vertexing, partcor)
TR011_LHC09a9ESD	TR011: ESD+MC analysis -> AOD + delta AOD + histograms	Completed	94%	📁	📁	154	146			TR011: ESD+MC analysis -> AOD + delta AOD + histograms
TR010_LHC09a7ESD	TR010: ESD+MC analysis -> AOD + delta AOD + histograms	Completed	85%	📁	📁	903	773			TR010: ESD+MC analysis -> AOD + delta AOD + histograms
TR009_LHC09a3ESD	TR009: ESD+MC analysis -> AOD + delta AOD + histograms	Completed	97%	📁		1087	1059			TR009: ESD+MC analysis -> AOD + delta AOD + histograms
TR008_LHC09a2ESD	TR008: ESD+MC analysis -> AOD + delta AOD + histograms	Completed	97%	📁	📁	1139	1108			TR008: ESD+MC analysis -> AOD + delta AOD + histograms

`http://pcalimonitor.cern.ch/train_details.jsp`

Bugs, problems, requests

https://savannah.cern.ch/bugs/?group=pdc



(+) Display Criteria

12 matching items - Items 1 to 12

Item ID ⇧	Summary	Submitted On	Assigned To	Submitted By
#63129	Request to produce AODs+deltas for LHC09a5 with AliRoot v4-19-01-AN	2010-02-18 20:39	None	dainesea
#60521	Femto task Configuration for real data	2009-12-11 16:42	None	akisiel
#60424	Fix AliFMDAnalysisTaskBackgroundCorrection::Terminate()	2009-12-10 08:22	hdalsgaa	agheata
#60107	AliCFContainer::MakeSlice: memory leak warnings	2009-12-04 13:26	rvernet	mfloris
#59815	Consistency between ESD and AOD for V0 task analysis	2009-11-30 11:24	bhippolyt	bhippolyt
#59806	request for running PWG3Muon analysis tasks in the next train	2009-11-30 10:20	mgheata	arnaldi
#59430	Feature Request: AliCFContainer::SetBinLabel(ivar,ibin,label)	2009-11-24 10:17	rvernet	mfloris
#59373	AliCFTrackQualityCuts::SelectionBitMap called with mctrack	2009-11-23 14:52	rvernet	mfloris
#59321	FORWARD merging of outputs	2009-11-23 10:13	hdalsgaa	mgheata

The Basics:
What the Analysis framework does in ALICE

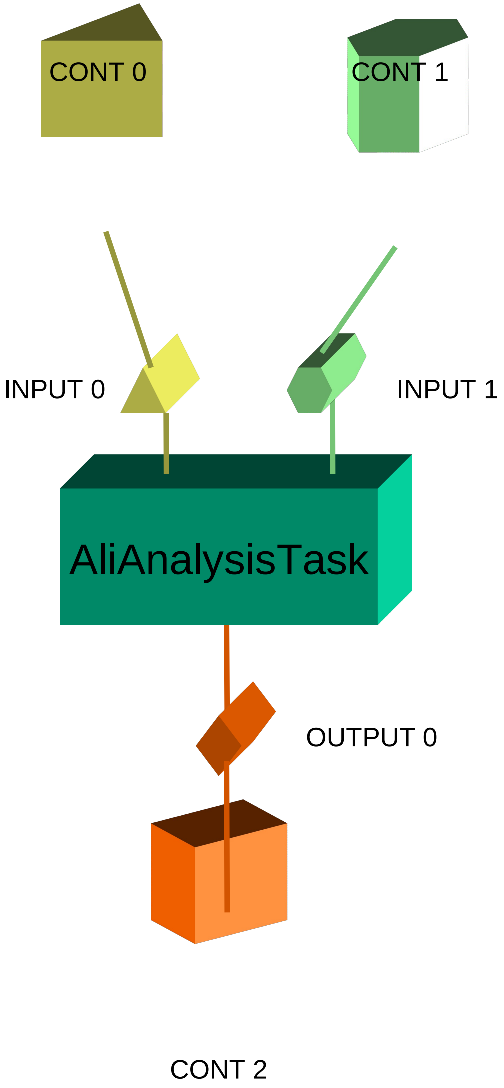
- Transparent access to all resources with the same code
  - Usage: Local, AliEn grid, CAF/PROOF
- Transparent access to different inputs
  - ESD, AOD, Kinematics tree (MC truth)
- Allow for „scheduled“ analysis
  - Common and well tested environment to run several tasks
- Defines a common terminology

N.B.: The analysis framework itself has a very general design, not bound to ALICE software

The single task view

- AliAnalyisTask
  - User provided code
- Input data
  - Provided via numbered slots
  - Each slot connected to a data container of the corresponding type at run time
  - Content can be any TObject
  - “Handlers” handle data specific operations
- Output data
  - Communicated via one or more slots
  - Handlers e.g. for AOD output
  - Simpler output e.g. histograms
  - Output can be disk resident (file) or only memory resident (transient data)
- **Several of these tasks can be**

collected in the *manager*

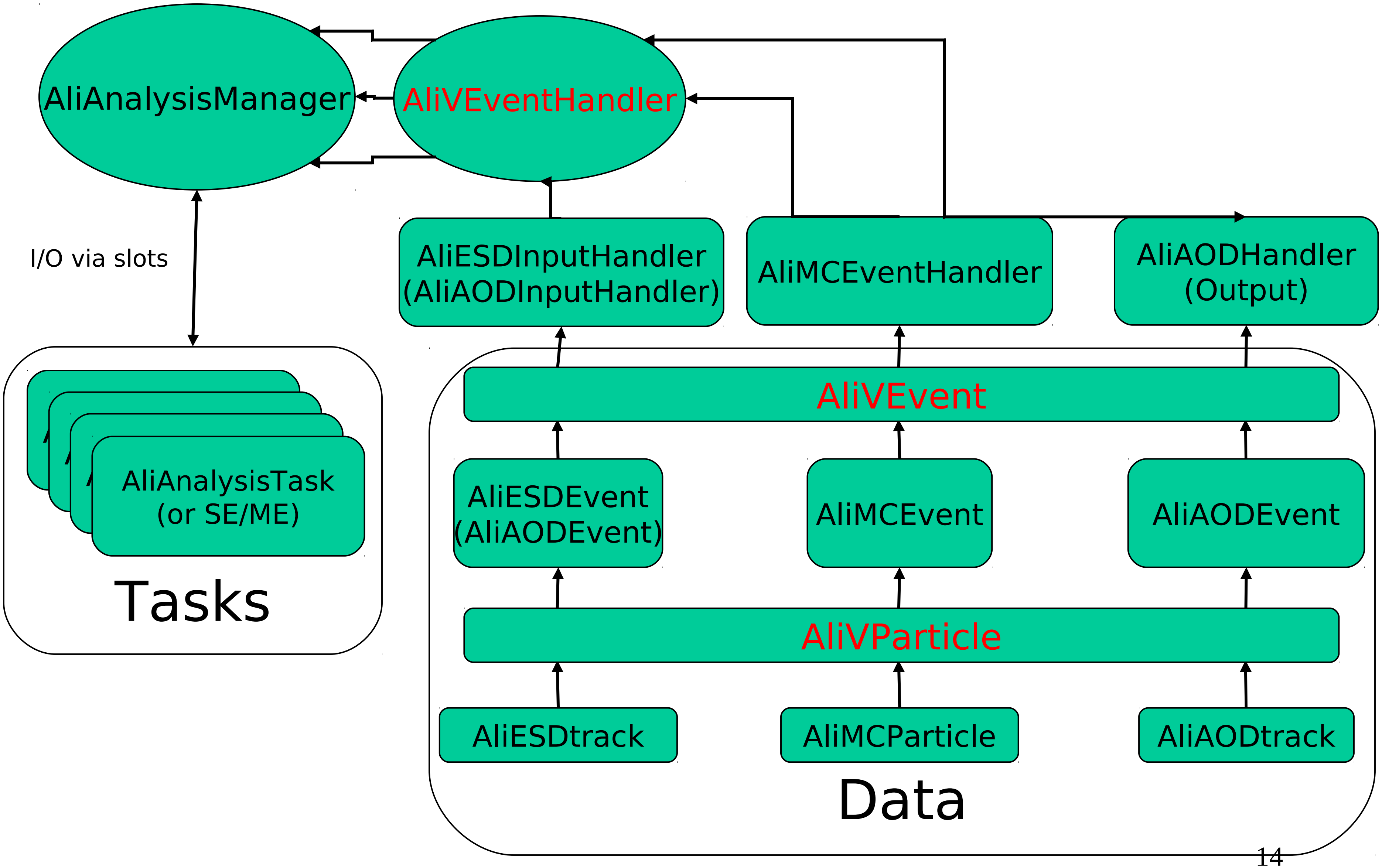


N:B.: `AliAnalysisTask` is a general Task
`AliAnalysisTaskSE` and `ME` are ALICE specific

Analysis manager

- Several analysis tasks registered to an analysis manager
  - Top data container(s) storing the initial input data chain (ESD, AOD, kinematics tree, ...)
  - Primary tasks executed in serial mode for each event
  - Possibly secondary tasks feeding from data produced by parent tasks
    - e.g. filtered AOD tracks, jets etc.
  - Handle initialization, execution and termination of all registered tasks

The overall picture



Library Structure

- ANALYSIS classes split
  - ALICE independent: libANALYSIS.so
  - ALICE specific: libANALAYSISalice.so
  - Load both in your steering macro
- Additional layer of inheritance
  - YourTask:AliAnalyisTaskSE(:AliAnalysisTask)
  - Does some work for you specific for “Single Event” analysis

An additional layer: AliAnalyisTaskSE

- Already provides the access to input ESD/AOD/Kinematics and define AOD output

```
AliVEvent*     fInputEvent;    //! VEvent Input
AliAODEvent*   fOutputAOD;     //! AOD out
AliMCEvent*    fMCEvent;       //! MC
```

- Input event can be cast as needed
  - AliESDEvent *esd = dynamic_cast<AliESDEvent*>fInputEvent;
- You need to implement at minimum:
  - **`virtual void UserCreateOutputObjects(){;}`**
  - **`virtual void UserExec(){;}`**
  - …instead of **`CreateOutputObjects()`**and **`Exec()`**
  - Have a look at:
    - analysis-tutorial.tgz#analysis-tutorial/TaskSE/

Analysis framework … in practice

Analysis development cycle

- In practice
  - Develop your analysis code as `AliAnalysisTaskSE` and test locally on a few files
    - Most debugging done here
  - When the code works locally, submit to PROOF or AliEn Grid
    - PROOF
      - Fast response, fast turnaround
      - Limited number of files
    - AliEn Grid: Access to all files
  - Optionally, add to the organized analysis

Analysis: Component by Component

- What is needed
  - The manager: *AliAnalysisManager*
  - The input handler: *AliESDInputHandler*
  - The output handler: *AliAODHandler*
- Optional
  - MC Truth handler:*AliMCEventHandler*
- Your Task(s) *AliAnalysisTask(SE)*
- A small execution macro
  - Load libraries
  - Collect input files (TChain), connect everything to the manager
  - Run

AliAnalysisManager

- AddTask(AliAnalysisTask *pTask)
  - At least 1 task per analysis (top task)
- CreateContainer(name, data_type, container_type, file_name)
  - Data can be optionally connected to a file
- ConnectInput/Output(pTask, islot, pContainer)
  - Mandatory for all data slots defined by used analysis modules
- InitAnalysis()
  - Performs a check for data type consistency and signal any illegal circular dependencies between modules
- StartAnalysis(const char *mode)
  - Starts the analysis in “local”, “proof” or “grid” mode

Analysis macro
Load libs and create manager

see analysis_tutorial.tgz#Task/jetana.C

```
// Load libs (more needed when running with root instead of
aliroot)
// Minimum need to load libANALYSIS
  gSystem->Load("libANALYSIS.so");
// AliAnalyTaskJets derives from AnalysisTaskSE:
  gSystem->Load("libANALYSISalice.so");

// Load the task
// AliAnalysisTaskJets is in libJETAN
   gSystem->Load("libJETAN.so");
// User tasks usually compiled on the fly, not needed here
//  gROOT->LoadMacro(”AliAnalysisMyTaskXYZ.cxx+g”);

// Create a Chain of input files ESDs here
// External file list is used
   gROOT->LoadMacro("$ALICE_ROOT/PWG0/CreateESDChain.C");
   TChain *chain = CreateESDChain("filelist.txt");
// or Manual chaining
// TChain *chain = new TChain("esdTree");
// chain->Add(”SomePath/AliESDs.root");

// Create the Analysis manager
  AliAnalysisManager *mgr  =
    new AliAnalysisManager("My Manager", "My Analysis");
```

Analysis macro
Create Input/Output handler

```
// Define Input Event Handler
  AliESDInputHandler* inpHandler = new AliESDInputHandler();

// Define Output Event Handler
  AliAODHandler* aodHandler = new AliAODHandler();
    aodHandler->SetOutputFileName("aod.root");

// Define MC Truth Event Handler
  AliMCEventHandler* mcHandler = new AliMCEventHandler();

// Add Handlers to the Task Manager
  mgr->SetInputEventHandler  (inpHandler);
  mgr->SetOutputEventHandler (aodHandler);
  mgr->SetMCtruthEventHandler(mcHandler);

// Be sure you are told what you are doing
  mgr->SetDebugLevel(10);
```

AOD or Kinematics Analysis?

- Same schema works for AOD analysis
  - TChain contains AOD files
  - User connects AliAODEvent to chain or retrieves it from AODInputHandler
- … and even for Kinematics
  - Add galice.root files to TChain

```
TChain *chain = new TChain(“TE”);
chain->Add(“/somePath/galice.root”);
```

  - This “triggers” correct loop over files
  - Obtain AliMCEvent from the manager combining
    - Kinematics tree
    - TreeE (Event Headers)
    - Track references

Analysis macro: Define Input/Output

```
// Declare Common Input TChain made of AliESDs.root files
  AliAnalysisDataContainer *cinput1 =
    mgr->GetCommonInputContainer();

// Common Output Tree in common output file AliAOD.root
// Not mandatory if task does not write AOD info
  AliAnalysisDataContainer *coutput1 =
    mgr->GetCommonOutputContainer();

// Private output objects writte to a file
  AliAnalysisDataContainer *coutput2 =
    mgr->CreateContainer("histos", TList::Class(),
         AliAnalysisManager::kOutputContainer, "histos.root");
```

AliAnalysisDataContainer

- Normally a class to be used ‘as is’
  - Enforcing a data type deriving from TObject
    - Type e.g. given by TChain::Class()
- Three types of data containers
  - Input – containing input data provided by AliAnalysisManager
  - Exchange – containing data transmitted between modules, or just to notify
  - Output – containing final output data of an analysis chain, eventually written to files.
- One can set a file name if the content is to be written

Analysis Macro
Add an Analysis Task and run

```
// Create Jet Finder Task task
  AliAnalysisTask *jetana = new
    AliAnalysisTaskJets("JetAnalysis");
  jetana->SetDebugLevel(10);

// Add task to the manager
  mgr->AddTask(jetana);

// Connect I/O to the task
  mgr->ConnectInput (jetana, 0, cinput1);
  mgr->ConnectOutput(jetana, 0, coutput1);
  mgr->ConnectOutput(jetana, 1, coutput2);

// Run the task
  mgr->InitAnalysis();
  mgr->PrintStatus();
  mgr->StartAnalysis(“local”,chain);
```

For jet analyis task see: $ALICE_ROOT/JETAN
AliAnalysisJets.{cxx,h}
JetAnalysisManagerLoc.C (more complex macro)

Implement Task: Method by Method

- We have a framework that calls an analysis task with inputs and outputs connected
- How do we implement our own analysis ask?
  - “Constructor” and “Destructor”
    - like any C++ class
  - UserCreateOutputObjects()
    - Create Histograms
  - UserExec()
    - The event loop
  - Terminate()
    - Called at the end, can draw e.g. a histogram
- We cover here the case for AliAnalysisTaskSE
  - Recommended to use TaskSE
  - Examples for AliAnalysisTask are in the tarball (Task/) for reference

AliAnalysisTaskSE

- Classes derived from AliAnalysisTaskSE can run locally, in PROOF and in AliEn

  - **"Constructor“*** called once on local PC
  - **UserCreateOutputObjects()**
  - **UserExec()** for each event
  - **Terminate()**

*Called in the macro

NB: The calling frequency is shown for LOCAL analysis, different in the PROOF case

Constructor:

```
AliAnalysisTaskJets::AliAnalysisTaskJets(const char*
name):
    AliAnalysisTaskSE(name),
    fConfigFile("ConfigJetAnalysis.C"),
    fNonStdBranch(""),
    fJetFinder(0x0),
    fHistos(0x0),
    fListOfHistos(0x0)
{
  DefineOutput(1, TList::Class()); // 0 slots assigned in
parent class
}
```

Called in the macro via new `AliAnalysisTaskJets(“JetAnalysis“)`

```
AliAnalysisTaskSE::AliAnalysisTaskSE(const char* name):…
{
   DefineInput (0,TChain::Class());
   DefineOutput(0,  TTree::Class());
}
```

Constructor:

```
AliAnalysisTaskJets::AliAnalysisTaskJets():
  AliAnalysisTaskSE(),
  fConfigFile("ConfigJetAnalysis.C"),
  fNonStdBranch(""),
  fJetFinder(0x0),
  fHistos(0x0),
  fListOfHistos(0x0)
{
  // Default constructor
}
```

N.B.: No DefineInput/DefineOutput in default c‘tor
(important for PROOF case)

Constructor

- User analysis module MUST derive from AliAnalysisTask
  - DefineInput/Output(Int_t islot, TClass *type)
  - Declared in the **named** class constructor
- Mandatory at least 1 input & 1 output slots
- For train operation: AliAnalysisTaskSE
  - Predefined input (TChain) and output (AOD)
  - Physics event selection
  - Background rejection

UserCreateOutputObjects()

```
// Open Histograms
   OpenFile(1);

   …
   fHisto = new TH1F(“fHisto”, “My Histo”, 100, 0., 10.);
   .......
 // Several histograms are more conveniently managed in a
TList
   fListOfHistos = new TList();
   fListOfHistos->Add(fHisto);
```

UserExec()

```
void AliAnalysisTaskJets::UserExec(Option_t */*option*/)
{
  // Execute analysis for current event
  //


  // Jet finding is delegated access to input outpout and
  // MC given by TaskSE
  fJetFinder->GetReader()->SetInputEvent(InputEvent(),
AODEvent(), MCEvent());
  fJetFinder->ProcessEvent();
  ….
  fHisto->Fill(pt);
  …
  // Post the data (it will be written automatically)
  PostData(1, fListOfHistos);
}
```

Called for each event

UserExec()

- Virtual void UserExec(Option_t *option)
  - Mandatory to implement in the derived class
  - This actually implements how the analysis module processes the current event from input data
    - End with PostData(slot, data) – will notify all tasks depending on the output that data is ready

Analysis framework: hands on

- Trivial example: plot the p_t of the ESD particles
- Files in the analysis_code.tgz archive (TaskSE)

```
AliAnalysisTaskPt.cxx
AliAnalysisTaskPt.h
files.txt
run1.C
```

Analysis framework: hands on

Have a look at run1.C

```
void run1()
  // load analysis framework
  gSystem->Load("libANALYSIS");
  gSystem->Load("libANALYSISalice");
  gROOT->LoadMacro("$ALICE_ROOT/PWG0/CreateESDChain.C");
  TChain* chain = CreateESDChain("files.txt", 2);

  // Create the analysis manager
  AliAnalysisManager *mgr = new AliAnalysisManager("testAnalysis");
  AliVEventHandler* esdH = new AliESDInputHandler;
  mgr->SetInputEventHandler(esdH);
  // Create task
  gROOT->LoadMacro("AliAnalysisTaskPt.cxx+g");
  AliAnalysisTask *task = new AliAnalysisTaskPt("TaskPt");
  // Add task
  mgr->AddTask(task);
  // Create containers for input/output
  AliAnalysisDataContainer *cinput = mgr->CreateContainer("cchain", TChain::Class(),
                                AliAnalysisManager::kInputContainer);
  AliAnalysisDataContainer *coutput = mgr->CreateContainer("chist", TH1::Class(),
                AliAnalysisManager::kOutputContainer,"Pt.ESD.1.root");
  // Connect input/output
  mgr->ConnectInput(task, 0, cinput);
  mgr->ConnectOutput(task, 0, coutput);
  // Enable debug printouts
  mgr->SetDebugLevel(2);

  if (!mgr->InitAnalysis())
    return;
  mgr->PrintStatus();
  mgr->StartAnalysis("local", chain);
}
```

Analysis framework: hands on

Have a look at AliAnalysisTaskPt.h

```
#ifndef AliAnalysisTaskPt_cxx
#define AliAnalysisTaskPt_cxx
class TH1F;

#include "AliAnalysisTaskSE.h"
class AliAnalysisTaskPt : public AliAnalysisTaskSE {
 Public:
  AliAnalysisTaskPt();
  AliAnalysisTaskPt(const char *name);
  virtual ~AliAnalysisTaskPt() {}


  virtual void   UserCreateOutputObjects();
  virtual void   UserExec(Option_t *option);
  virtual void   Terminate(Option_t *);

 private:
  TH1F        *fHistPt; //Pt spectrum
  ClassDef(AliAnalysisTaskPt, 1); // example of analysis
};
#endif
```

Analysis framework: hands on

```
//_____________________________________________________________________________
AliAnalysisTaskPt::AliAnalysisTaskPt(const char *name)
  : AliAnalysisTaskSE(name), fHistPt(0)
{
  // Constructor
  // Define input and output slots here
  // Slot #0 works are defined in TaskSE
  // Output slot #1 writes into a TH1 container
  DefineOutput(1, TH1F::Class());
}

//_____________________________________________________________________________
void AliAnalysisTaskPt::UserCreateOutputObjects()
{
  // Create histograms
  // Called once

  fHistPt = new TH1F("fHistPt", "P_{T} distribution", 15, 0.1, 3.1);
  fHistPt->GetXaxis()->SetTitle("P_{T} (GeV/c)");
  fHistPt->GetYaxis()->SetTitle("dN/dP_{T} (c/GeV)");
  fHistPt->SetMarkerStyle(kFullCircle);
}
```

Have a look at
AliAnalysisTaskPt.cxx

Only in the constructor
with the signature (const char *)

Analysis framework: hands on

```
void AliAnalysisTaskPt::UserExec(Option_t *)
{
  // Main loop
  // Called for each event
  AliVEvent *event = InputEvent();
  if (!event) {
     Printf("ERROR: Could not retrieve event");
     return;
  }

  if(Entry()==0){
    AliESDEvent* esd = dynamic_cast<AliESDEvent*>(event);
    AliAODEvent* aod = dynamic_cast<AliAODEvent*>(event);
    if(esd){
      Printf("We are reading from ESD");
    }
    else if(aod){
      Printf("We are reading from AOD");
    }
  }

  Printf("There are %d tracks in this event", event->GetNumberOfTracks());
  // Track loop to fill a pT spectrum
  for (Int_t iTrack = 0; iTrack < event->GetNumberOfTracks(); iTrack++) {
    AliVParticle *track = event->GetTrack(iTrack);
    if (!track) {
      Printf("ERROR: Could not receive track %d", iTrack);
      continue;
    }
    fHistPt->Fill(track->Pt());
  } //track loop

  // Post output data.
  PostData(1, fHistPt);
}
```

Works for AOD and ESD Input

Side Remark: MC truth

```
void AliAnalysisTaskXYZ::UserExec(Option_t* option )
{

  // During Analysis
  AliVEvent* mc = MCEvent();
  Int_t ntrack = mc->GetNumberOfTracks();
  for (Int_t i = 0; i < ntrack; i++)
  {
    AliVParticle* particle = mc->GetTrack(i);
    Double_t pt = particle->Pt();
  }

}
```

Can also read only Kinematics (no need for ESDs), without ESDs change one line in steering macro:

```
chain = CreateChain(“TE”,galice_root_list,2);
```

Analysis framework: hands on

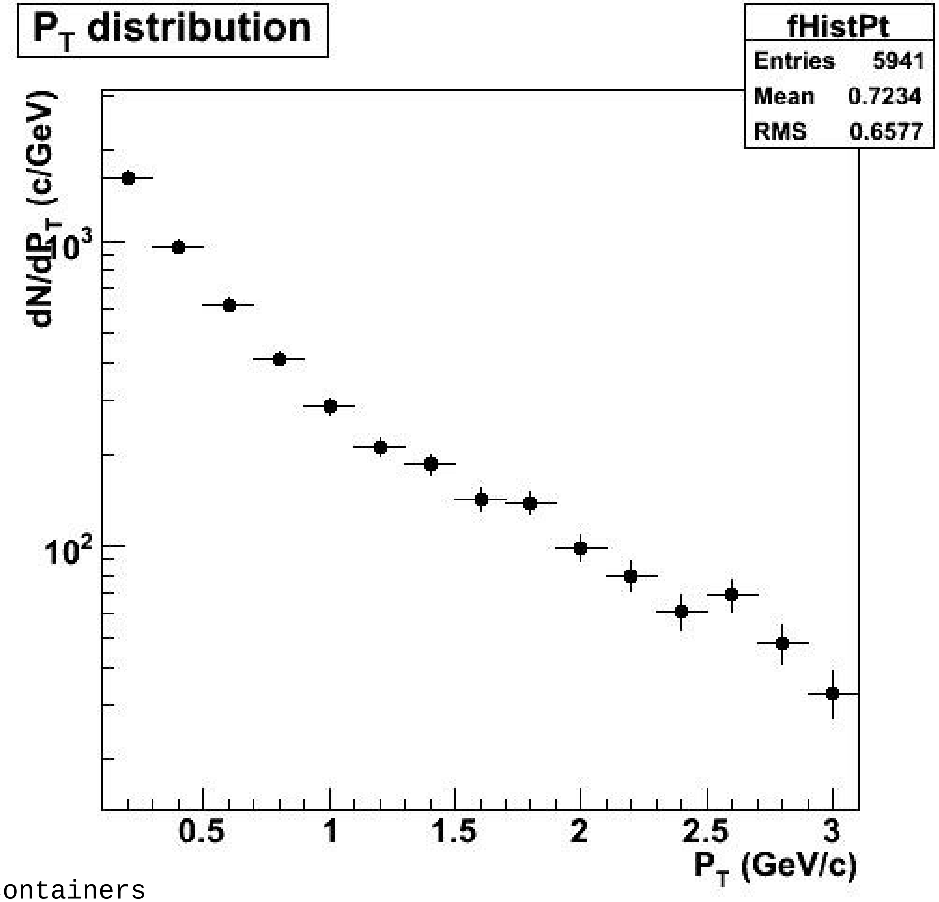
```
void AliAnalysisTaskPt::Terminate(Option_t *)
{
  // Draw result to the screen
  // Called once at the end of the query

  fHistPt =
dynamic_cast<TH1F*>(GetOutputData(1));
  if (!fHistPt) {
    Printf("ERROR: fHistPt not available");
    return;
  }

  TCanvas *c1 = new
TCanvas("AliAnalysisTaskPt","Pt",10,10,510,510);
  c1->cd(1)->SetLogy();
  fHistPt->DrawCopy("E");
}
```

Et voila...

```
root[0].x run1.C
Processing run1.C...
task: TaskPt  ACTIVE=0 POST_LOOP=0
   INPUT #0: TChain <-  [cchain]
   OUTPUT #0: TH1F ->  [chist]
   Container: chist  type: TH1 POST_LOOP=0
 = Data producer: task TaskPt    = Consumer tasks: -none-
Filename: Pt.ESD.1.root
StartAnalysis: testAnalysis
===== RUNNING LOCAL ANALYSIS testAnalysis ON TREE esdTree
->AliAnalysisSelector->Init: Analysis manager restored
->AliAnalysisSelector->SlaveBegin() after Restore
->AliAnalysisManager::SlaveBegin()
->AliAnalysisManager::Init(esdTree)
<-AliAnalysisManager::Init(esdTree)
<-AliAnalysisManager::SlaveBegin()
<-AliAnalysisSelector->SlaveBegin()
AliAnalysisManager::Notify() file: AliESDs.root
->AliAnalysisSelector::Process()
== AliAnalysisManager::GetEntry()
AliAnalysisManager::ExecAnalysis
 Executing task TaskPt
…
<-AliAnalysisSelector::Process()
->AliAnalysisSelector::SlaveTerminate()
->AliAnalysisManager::PackOutput()
<-AliAnalysisManager::PackOutput: output list contains 0 containers
<-AliAnalysisSelector::SlaveTerminate()
->AliAnalysisSelector::Terminate()
->AliAnalysisManager::UnpackOutput()
   Source list contains 0 containers
<-AliAnalysisManager::UnpackOutput()
->AliAnalysisManager::Terminate()
<-AliAnalysisManager::Terminate()
<-AliAnalysisSelector::Terminate()
```



Hands on: Excercises

- Replace `AliAnalysisTaskPt` by `AliAnalysisTaskPtMC` in `run1.C`
  - What happens?
  - Why?
  - How to fix it?
- Try to run `AliAnalysisTaskPt` and `AliAnalysisTaskPtMC` together
- Try to run with root alone instead of aliroot

Some extras….

Collision Event Selection

Mixed Events and the
Correction Framework

Collision Event Selection for 2009 Runs

- Use the analysis framework
- Derive from AliAnalysisTaskSE
- Use input output slots > 0
- Two new classes
  - *AliPhysicsSelection* (Jan Fiete)
  - *AliBackgroundSelection* (Michele)
- Integrated into framework, via *AliPhysicsSelectionTask*
  - Delegates the selection to *AliESDInputHandler* and *AliAnalysisTaskSE*
  - Produces bookkeeping histograms

Add the task *AliPhysicsSelectionTask* using:

```
gROOT->LoadMacro("$ALICE_ROOT/ANALYSIS/macros/AddTaskPhysicsSelection.C");
AliPhysicsSelectionTask* physSelTask = AddTaskPhysicsSelection();
```

By default the physics selection for real data is activated and background rejection is performed using the class *AliBackgroundSelection*.

You can also run the selection on Monte Carlo Events using the first argument of *AddTaskPhysicsSelection*.

```
AliPhysicsSelectionTask* physSelTask = AddTaskPhysicsSelection(kTRUE)
```

With the second argument you can switch off background rejection. Here the example for real data and background rejection switched off:

```
AliPhysicsSelectionTask* physSelTask = AddTaskPhysicsSelection(kFALSE, kFALSE)
```

To activate the selection for your task (works for tasks deriving from *AliAnalysisTaskSE*):

```
yourTask->SelectCollisionCandidates();
```

In this case your *UserExec()* will be called only for selected events. Alternatively, you can use the result of the selection inside your task with the following line:

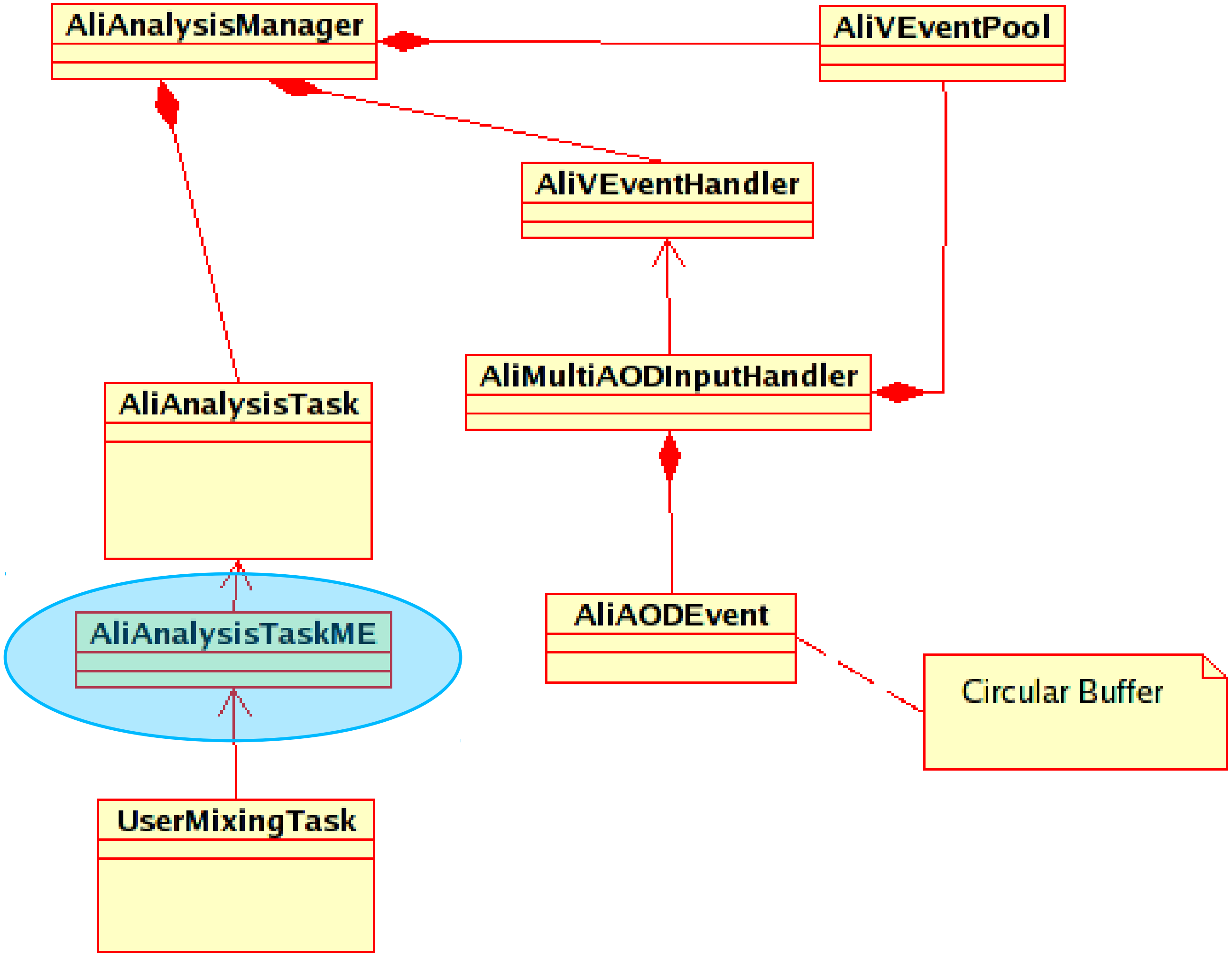
```
Bool_t isSelected = ((AliInputEventHandler*)(AliAnalysisManager::GetAnalysisManager()->GetInputEventHandler()))->IsEventSelected();
```

Mixed Events

- Implemented since v4-13-Release
- Needed for any analysis that suffers from combinatorial background
  - e.g. photon pair combinations for π^0 -> $\gamma\gamma$ analysis
- Inherit from `AliAnalysisTaskME`
  - Provides access to a pool of events which are “close” (e.g. in multiplicity) to the current event (tags needed for selction)
  - Pool stored independent of user requirements only once.

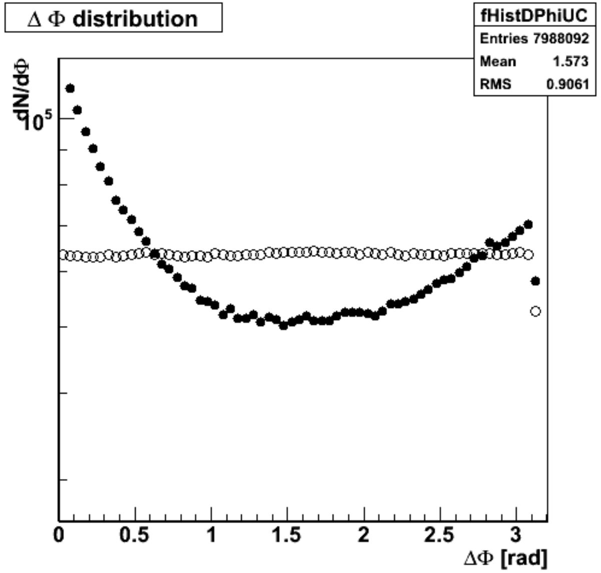
Mixed Events

Analysis With Event Mixing



Example Mixed Events

- $ALICE_ROOT/ANALYSIS/ (trunk)
  - AliAnalysisTaskPhiCorr.{cxx,h}
  - DphiAnalysis.C
  - Data from /afs/cern.ch/user/m/morsch/public/

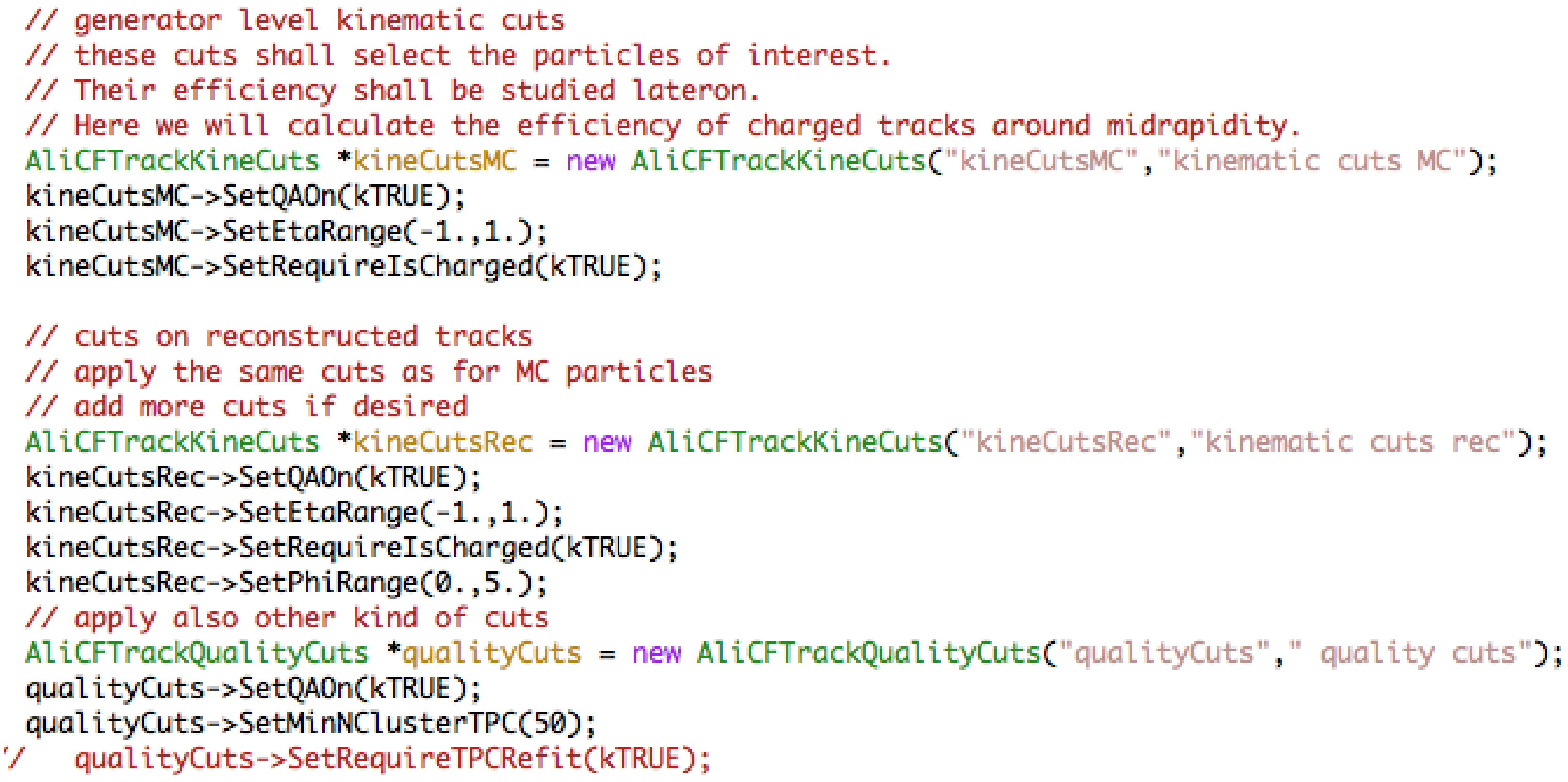


Correction framework

- In general efficiency is:
  - Output(x_1,x_2,x_3...)/Input(x_1,x_2,x_3...)
  - (x_1,x_2,x_3...) e.g. (p_T,η,z,...)
- $ALICE_ROOT/CORRFW provides
  - Container classes
    - n-dim histograms which store distributions with
      - MC input
      - after acceptance cut (with track references)
      - reconstrcuted tracks w/wo cuts
  - Cut classes
- Basic example:
  - analysis-tutorial.tgz#analysis-tutorial/TaskSE/

Correction Framework Crash Course

Create the Cuts (see run6.C)

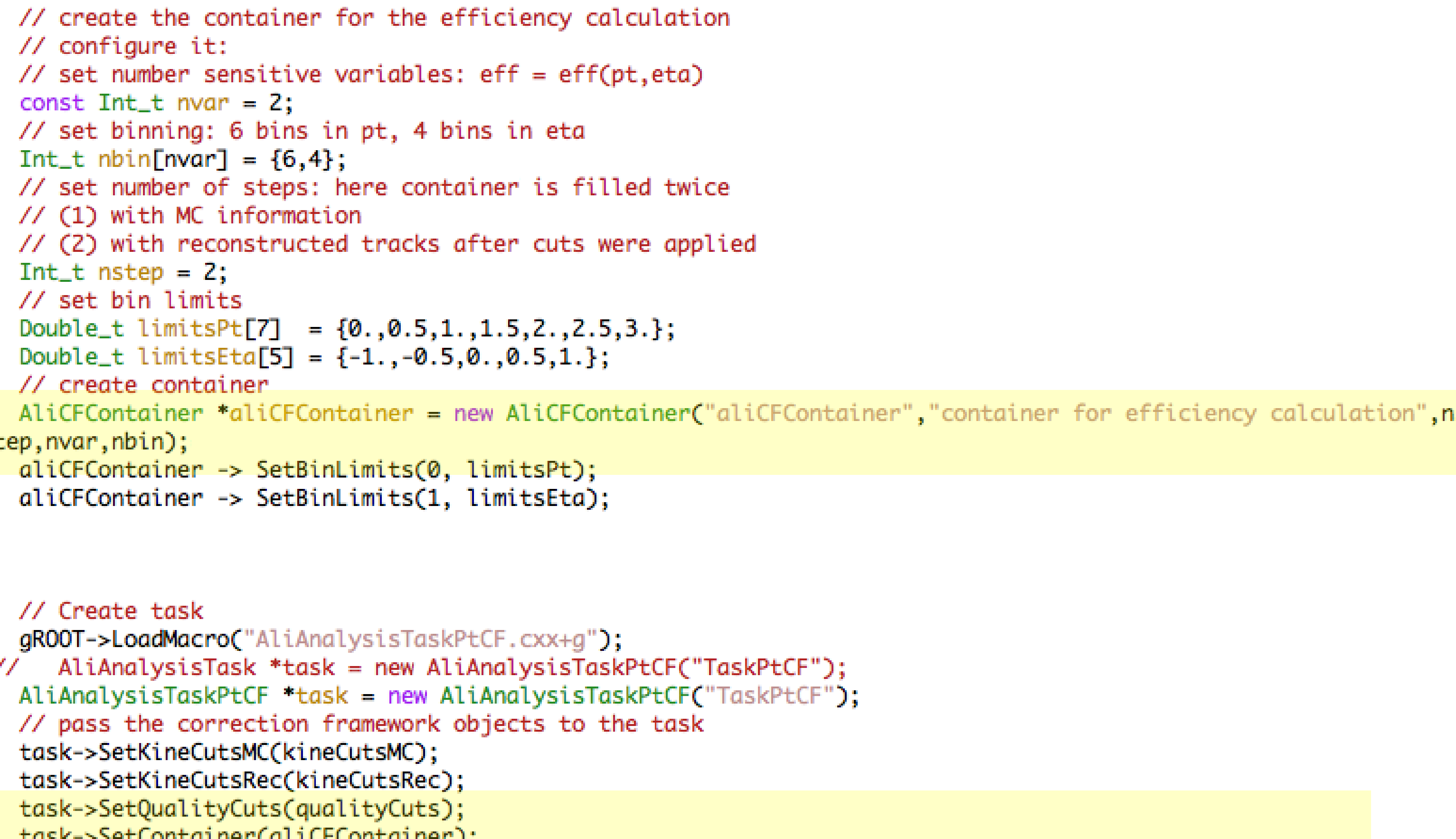


```
// generator level kinematic cuts
// these cuts shall select the particles of interest.
// Their efficiency shall be studied lateron.
// Here we will calculate the efficiency of charged tracks around midrapidity.
AliCFTrackKineCuts *kineCutsMC = new AliCFTrackKineCuts("kineCutsMC","kinematic cuts MC");
kineCutsMC->SetQAOn(kTRUE);
kineCutsMC->SetEtaRange(-1.,1.);
kineCutsMC->SetRequireIsCharged(kTRUE);

// cuts on reconstructed tracks
// apply the same cuts as for MC particles
// add more cuts if desired
AliCFTrackKineCuts *kineCutsRec = new AliCFTrackKineCuts("kineCutsRec","kinematic cuts rec");
kineCutsRec->SetQAOn(kTRUE);
kineCutsRec->SetEtaRange(-1.,1.);
kineCutsRec->SetRequireIsCharged(kTRUE);
kineCutsRec->SetPhiRange(0.,5.);
// apply also other kind of cuts
AliCFTrackQualityCuts *qualityCuts = new AliCFTrackQualityCuts("qualityCuts"," quality cuts");
qualityCuts->SetQAOn(kTRUE);
qualityCuts->SetMinNClusterTPC(50);
//   qualityCuts->SetRequireTPCRefit(kTRUE);
```

Correction Framework Crash Course

Create the Containers and create the Task (see run6.C)



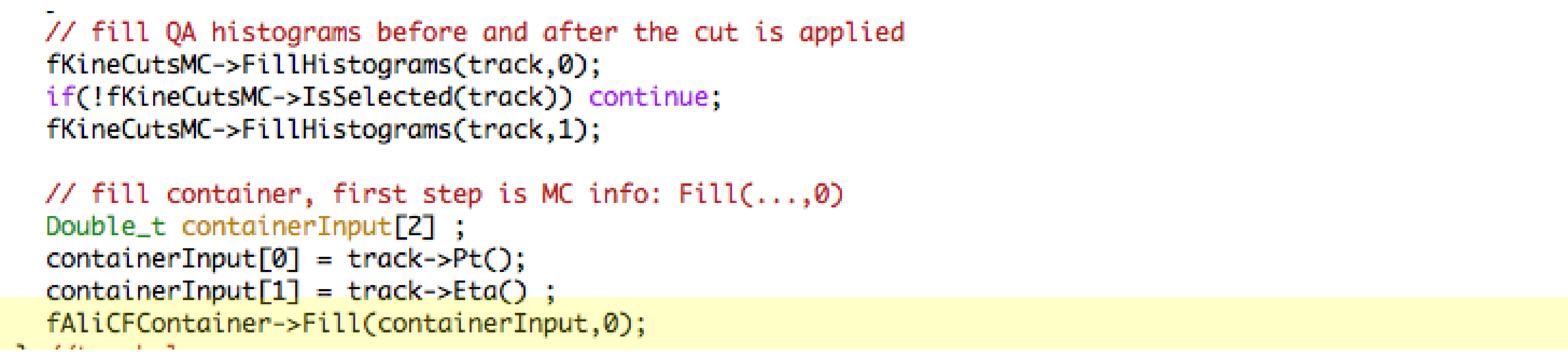
```
// create the container for the efficiency calculation
// configure it:
// set number sensitive variables: eff = eff(pt,eta)
const Int_t nvar = 2;
// set binning: 6 bins in pt, 4 bins in eta
Int_t nbin[nvar] = {6,4};
// set number of steps: here container is filled twice
// (1) with MC information
// (2) with reconstructed tracks after cuts were applied
Int_t nstep = 2;
// set bin limits
Double_t limitsPt[7]  = {0.,0.5,1.,1.5,2.,2.5,3.};
Double_t limitsEta[5] = {-1.,-0.5,0.,0.5,1.};
// create container
AliCFContainer *aliCFContainer = new AliCFContainer("aliCFContainer","container for efficiency calculation",n
tep,nvar,nbin);
aliCFContainer -> SetBinLimits(0, limitsPt);
aliCFContainer -> SetBinLimits(1, limitsEta);



// Create task
gROOT->LoadMacro("AliAnalysisTaskPtCF.cxx+g");
//   AliAnalysisTask *task = new AliAnalysisTaskPtCF("TaskPtCF");
AliAnalysisTaskPtCF *task = new AliAnalysisTaskPtCF("TaskPtCF");
// pass the correction framework objects to the task
task->SetKineCutsMC(kineCutsMC);
task->SetKineCutsRec(kineCutsRec);
task->SetQualityCuts(qualityCuts);
task->SetContainer(aliCFContainer);
```

Correction Framework Crash Course

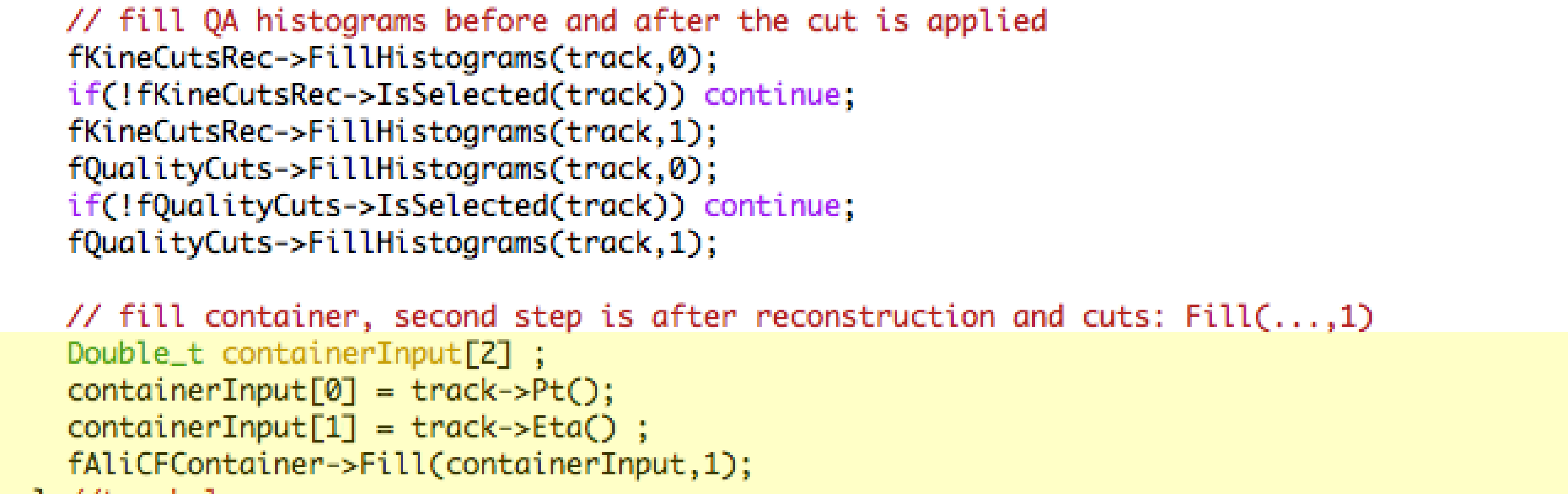
Fill MC information in AliAnaylisTaskCF::UserExec() after kinematical cuts (see AliAnaylisTaskCF.cxx)



```
// fill QA histograms before and after the cut is applied
fKineCutsMC->FillHistograms(track,0);
if(!fKineCutsMC->IsSelected(track)) continue;
fKineCutsMC->FillHistograms(track,1);

// fill container, first step is MC info: Fill(...,0)
Double_t containerInput[2] ;
containerInput[0] = track->Pt();
containerInput[1] = track->Eta() ;
fAliCFContainer->Fill(containerInput,0);
```

Fill reconstructed information after QA cuts

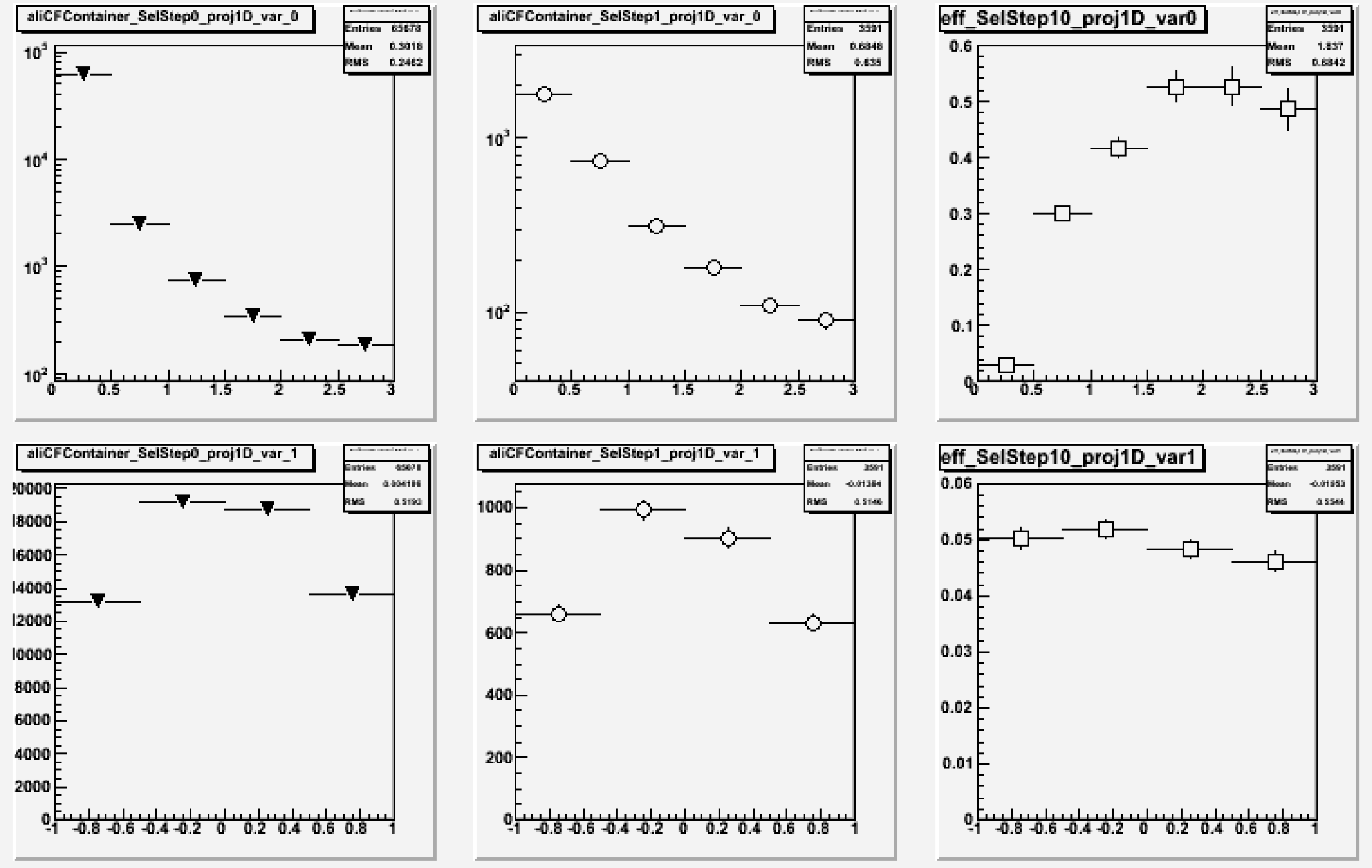


```
// fill QA histograms before and after the cut is applied
fKineCutsRec->FillHistograms(track,0);
if(!fKineCutsRec->IsSelected(track)) continue;
fKineCutsRec->FillHistograms(track,1);
fQualityCuts->FillHistograms(track,0);
if(!fQualityCuts->IsSelected(track)) continue;
fQualityCuts->FillHistograms(track,1);

// fill container, second step is after reconstruction and cuts: Fill(...,1)
Double_t containerInput[2] ;
containerInput[0] = track->Pt();
containerInput[1] = track->Eta() ;
fAliCFContainer->Fill(containerInput,1);
```

Try it Out

- run TaskCF/run6.C and TaskCF/CalcEff_run6.C

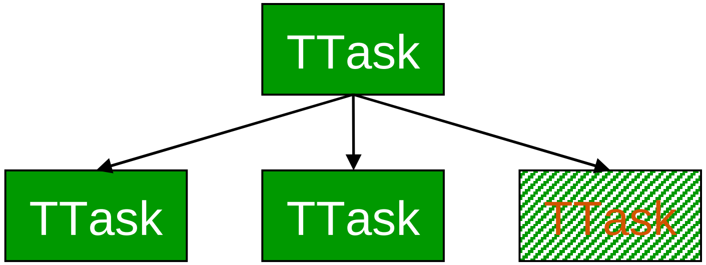


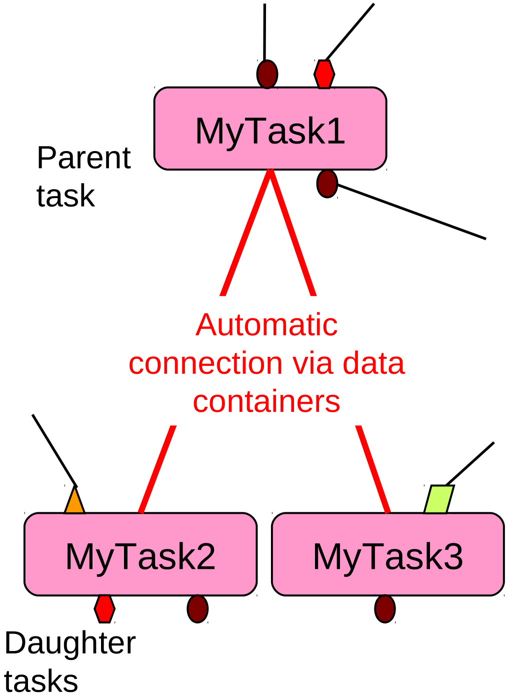
To be continued…

Analysis Framework …technicalities (Backup)

Advanced Structure

- Analysis has to be split in functional modules
  - At least one
  - Deriving from TTask
  - Parent task running active daughters
- Modules are not manually inter-connected
  - Connected just to input/output data containers
  - A data container has one provider and possibly several clients
  - A module becomes active when all input data is ready





AliAnalysisDataContainer

- Normally a class to be used ‘as is’
  - Enforcing a data type deriving from TObject
  - For non-TObject (e.g. basic) types one can subclass and append the needed types as data members
- Three types of data containers
  - Input – containing input data provided by AliAnalysisManager
  - Transient – containing data transmitted between modules
  - Output – containing final output data of an analysis chain, eventually written to files.
- One can set a file name if the content is to be written

AliAnalysisTask::ConnectInputData( )

```
// Get the input handler from the manager
  AliESDInputHandler* esdH = (AliESDInputHandler*)
    ((AliAnalysisManager::GetAnalysisManager()
      ->GetInputEventHandler());

// Get pointer to esd event from input handler
  AliESDEvent* fESD = esdH->GetEvent();
```

- EsdFilter
- default constructor
- copy from afs area
- check corrfw
- clean up some slides
- which aliroot version?
- refer to analysis train example

References

- **This tutorial:**
  - **https://aliceinfo.cern.ch/Offline/AliRoot/Manual.html**
- **Analysis web pages:**
  - **http://aliceinfo.cern.ch/Offline/Activities/Analysis/**
- **Analysis framework:**
  - **http://aliceinfo.cern.ch/Offline/Activities/Analysis/AnalysisFra**
- **Analysis train:**
  - **http://aliceinfo.cern.ch/Offline/Activities/Analysis/AnalysisFra**
- **News and known problems RSS:**
  - **http://aliceinfo.cern.ch/Offline/Activities/Analysis/NewsAndPro**
- **Analysis task force mailing list:**
  - **alice-project-analysis-task-force@cern.ch**