ANALYSIS FRAMEWORK IN ALICE

Andrei Gheata

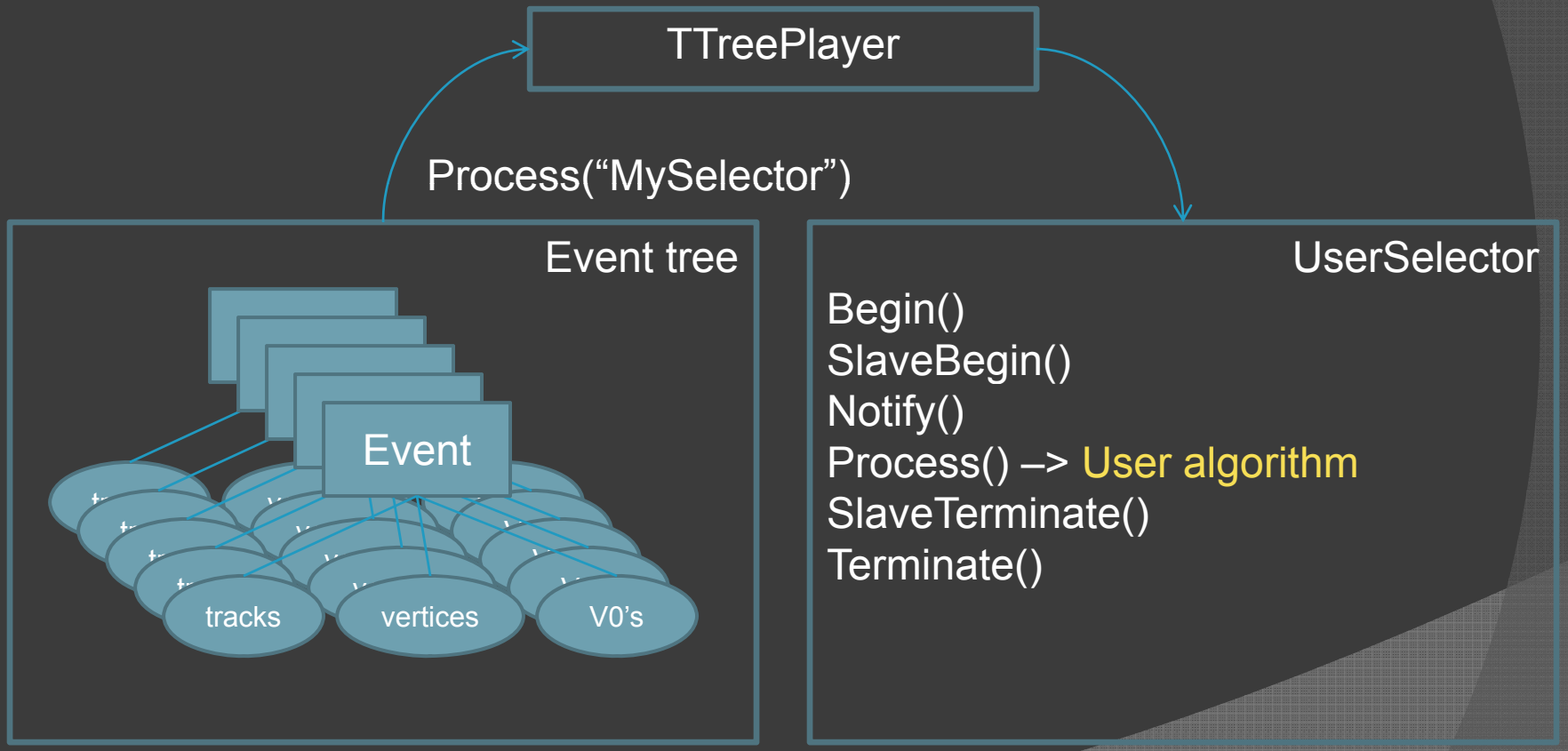
Analysis in ALICE

- Three main analysis modes
  - Prompt data processing @CERN with PROOF
    - calibration, alignment, reconstruction, analysis
  - Analysis with local PROOF clusters
  - Batch Analysis on the GRID infrastructure
- Scheduled analysis (see talk of Andreas)
  - At least at the beginning
  - Needed for optimizing CPU/IO over large datasets and very sparse resources

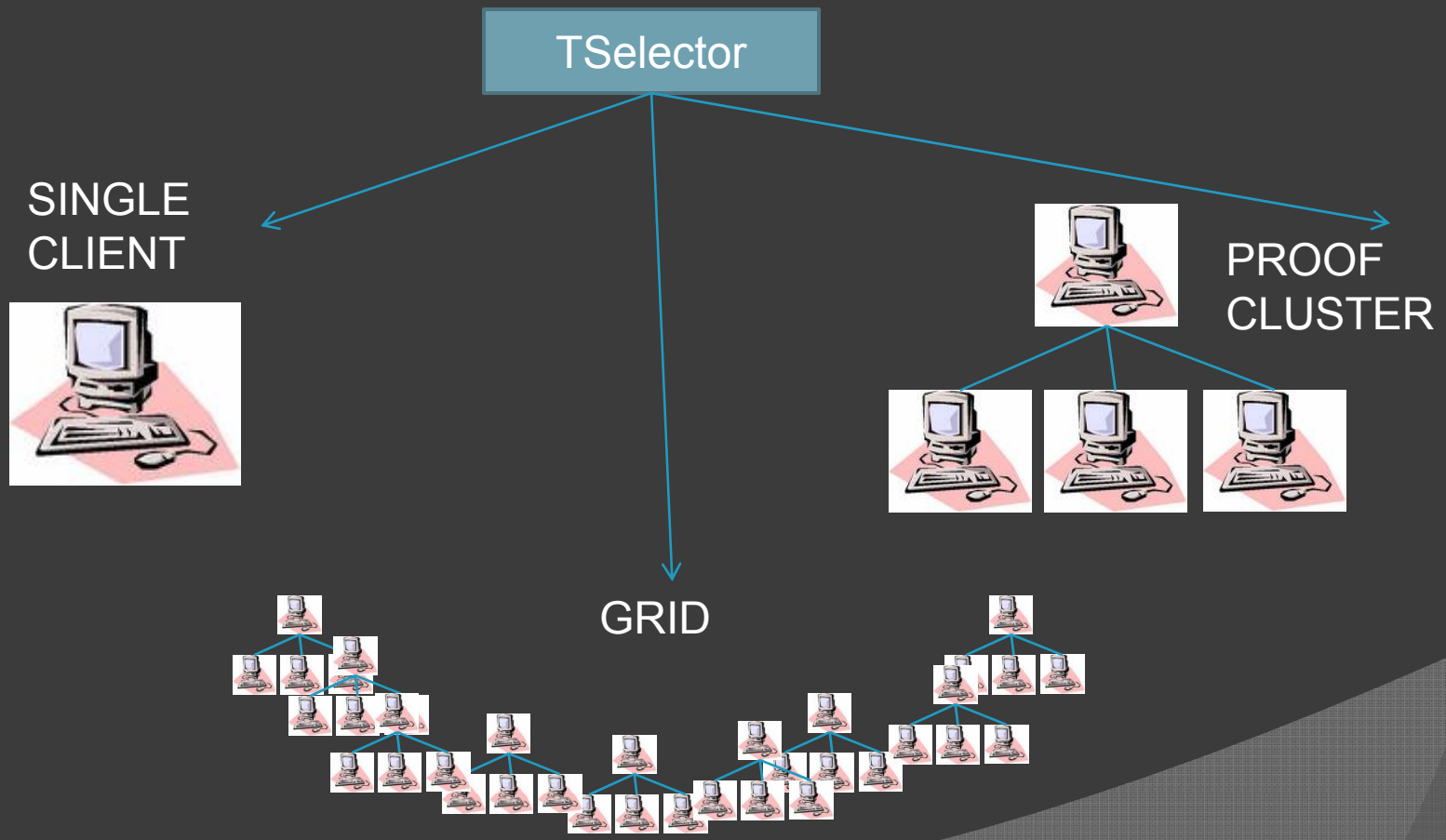
Main goals for an analysis framework in ALICE

- Provide transparent access to all resources with the same code
  - Usage: Local, AliEn grid, CAF/PROOF
- Provide ransparent access to different inputs
  - ESD, AOD, Kinematics tree (MC truth)
- Allow sharing resources by multiple analysis modules
  - Framework must allow accommodating different analysis in the same session

TSelector – an event loop handler for trees



TSelector – an event loop handler for trees



FRAMEWORK

Analysis as a task

- Selector model - simple user analysis looping input events and producing some histograms
- Need to federate several different analysis algorithms
  - Sharing the same input data
  - Using data produced by a different module
    - Data flow within analysis chains
- Give a maximum of workload to CPU while data is in memory

From selectors to tasks

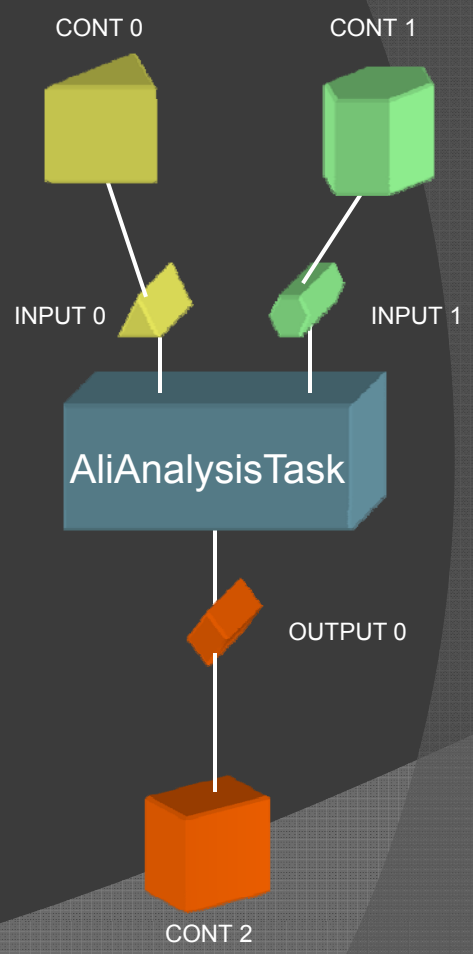
- Base class providing user methods called by the framework at well-defined stages (very close to TSelector ones)
- Analysis modules may connect one to another in a workflow
  - E.g ESD -> filtering -> AOD -> PWG -> histograms
  - ROOT TTask well suitable
- Schema needed for making it work -> **Analysis framework**

UserTask : public AliAnalysisTask

- ConnectInputData()
  - Define which data is connected to which slot
- CreateOutputObjects()
  - Create Histograms
- Init(),LocalInit()
  - Optional, e.g. read parameters
- Exec()
  - The event loop
- Terminate()
  - Called at the end, can draw e.g. a histogram

AliAnalysis... Framework

- Data-oriented model composed of independent tasks
  - Task execution triggered by data readiness
- Tasks are owned and managed by *AliAnalysisManager*
- Parallel execution and event loop done via *TSelector* functionality
  - Mandatory for usage with PROOF

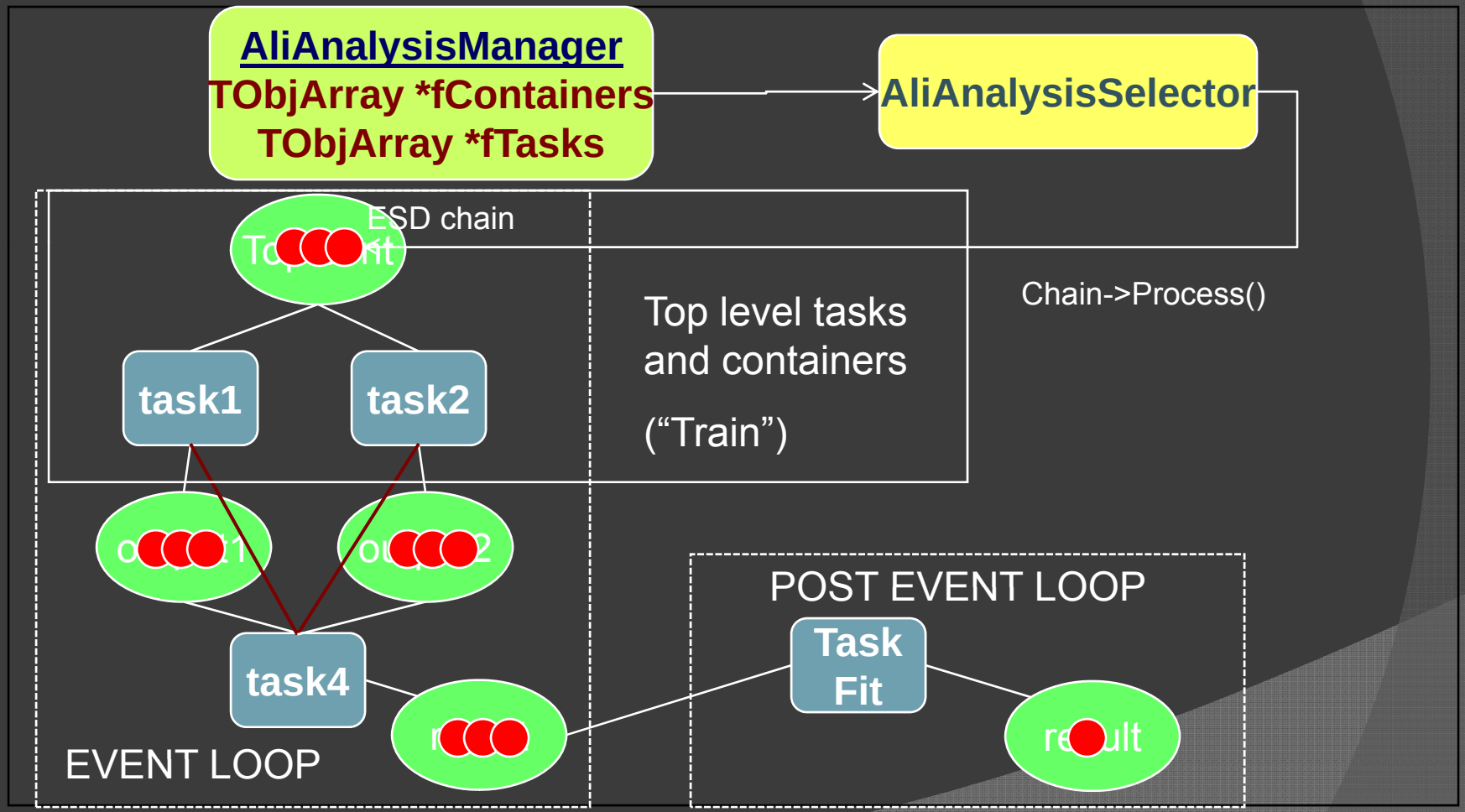


N.B.: The analysis framework itself has a very general design, not bound to ALICE software

Why a common framework

- To make data processing more efficient
  - In particular if resources are sparse.
  - Optimise CPU/IO ratio
  - Make possible running user analysis in PROOF or GRID, but hide as much as possible these technologies (same user code)
- But also
  - Develops common knowledge base and terminology.
  - Helps documenting the analysis procedure and makes results reproducible.

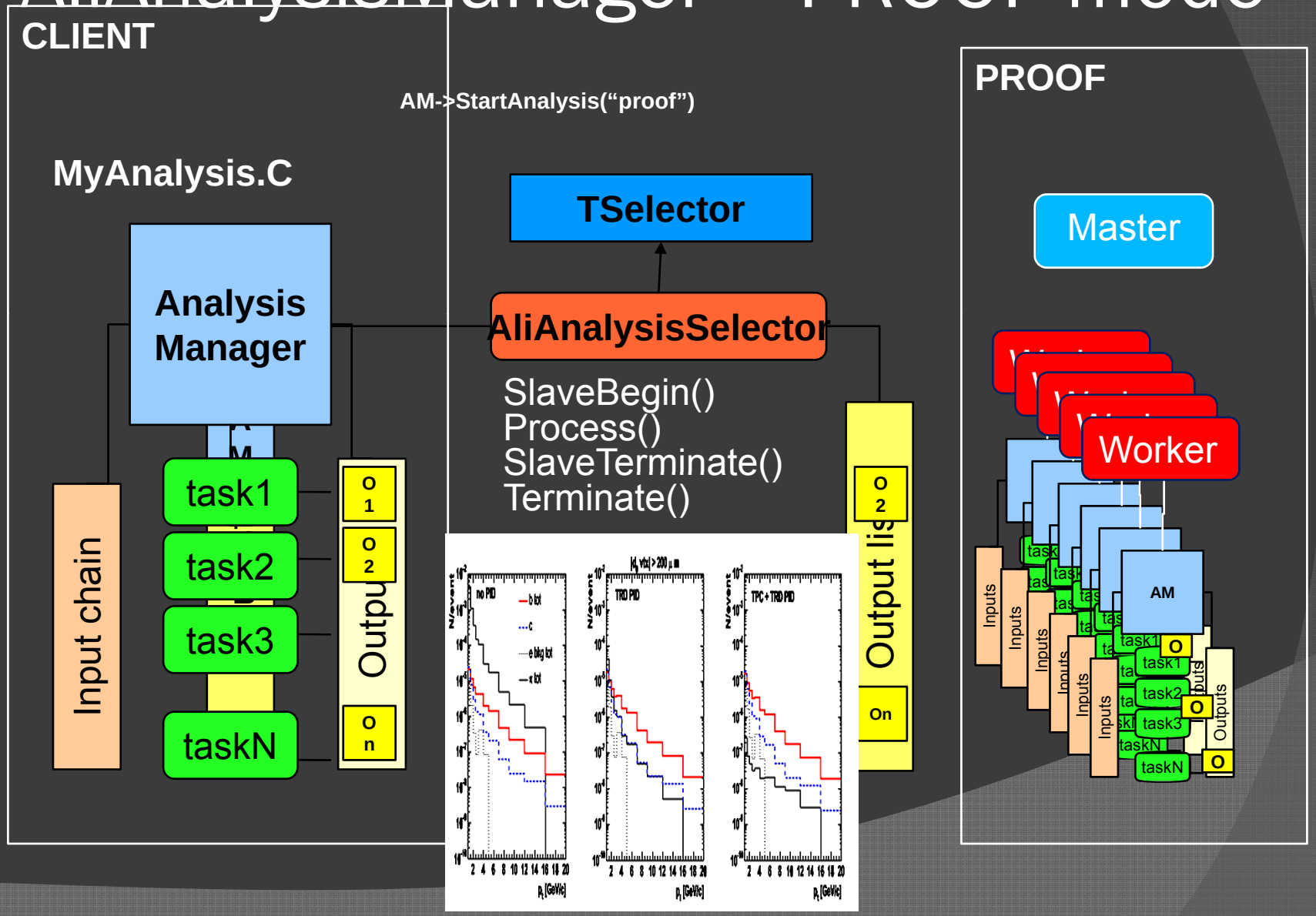
Tasks and event loop



AliAnalysisManager

- Holds lists of user-defined tasks and their data containers
- Provides simple user interface to connect tasks and containers in a working chain
- Spawns a custom TSelector to initiate the event loop over the input chain
- Task event processing delegated by AliAnalysisSelector to the manager

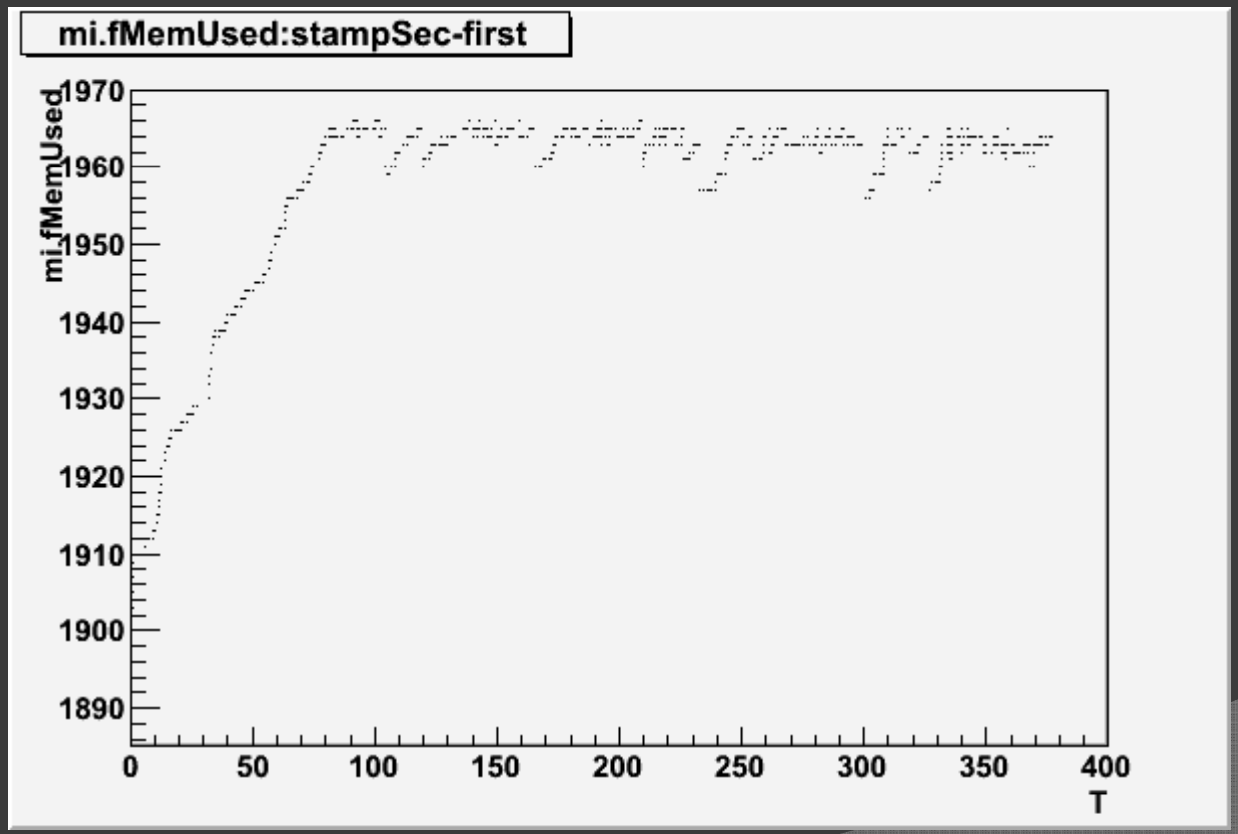
AliAnalysisManager – PROOF mode



Memory checking

- **AliAnalysisManager::SetNSysInfo(Long64_t nevents)**
- Monitoring memory&timing each N events
  - Using AliSysInfo functionality to spot leaking tasks
  - Only in local mode
  - 2 output files: syswatch.log and syswatch.root
- Currently investigating how to improve monitoring per task

Memory usage profile vs time



PROOF-related add-ons

- Processing datasets instead of a chain

StartAnalysis(const char *type, const char *dataset,…)

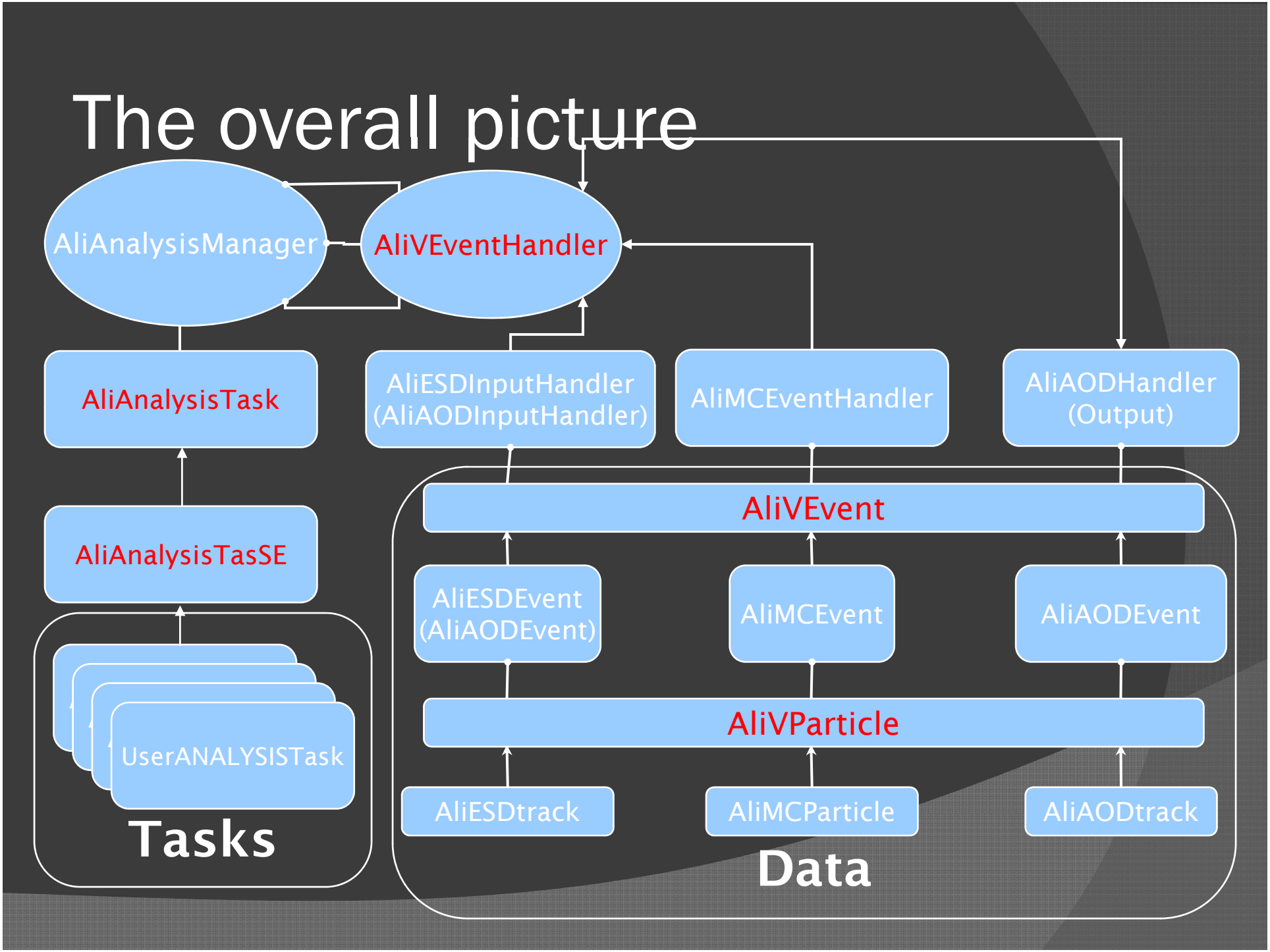
- Saving files on proof workers

AliAnalysisDataContainer::SetSpecialOutput()

AliAnalysisManager::SetSpecialOutputLocation(const char* dir)

  - Specified output files will be written to the special output location using TFile::Cp()
  - If location not specified, output files are merged by PROOF and registered or sent back to client

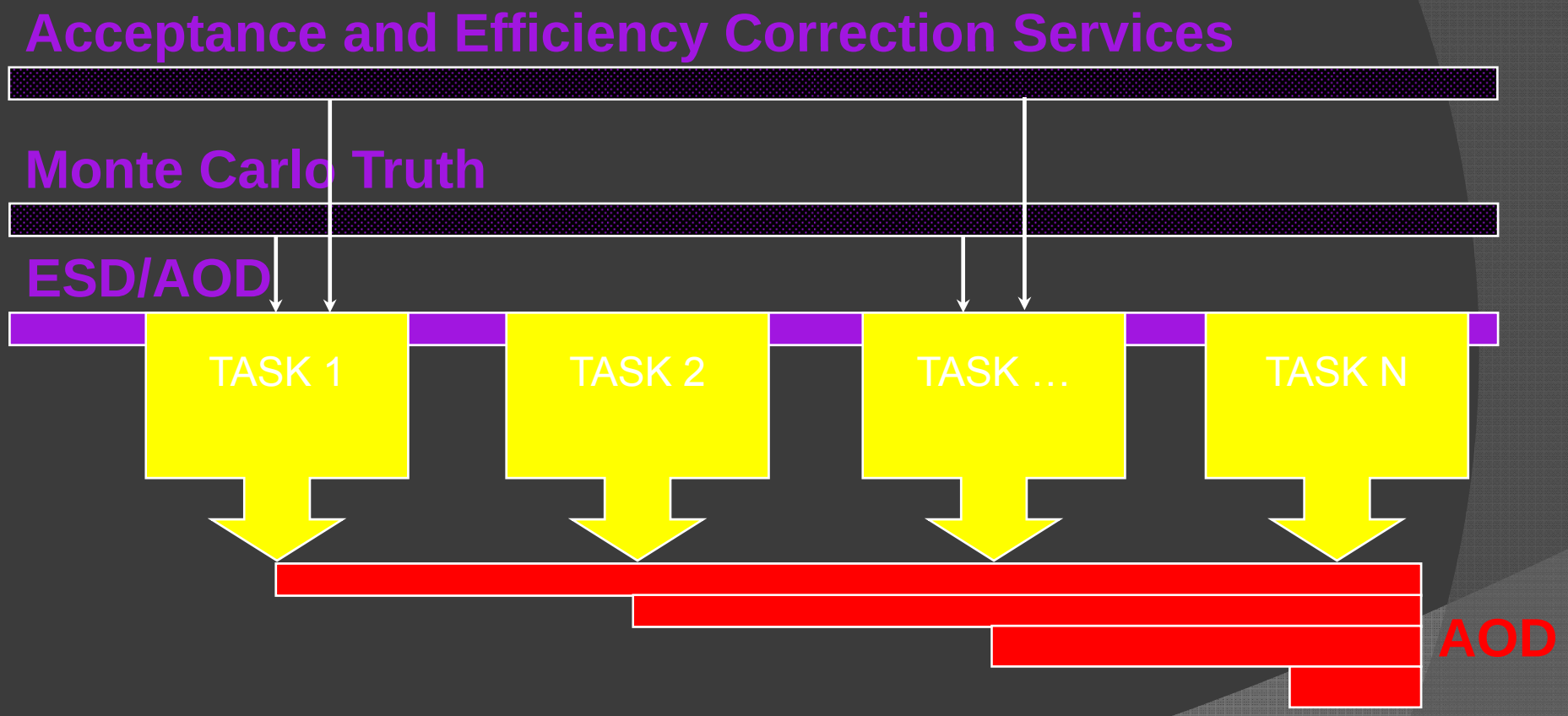
The overall picture



Integration of User Tasks

- Relatively smooth so far
- Needs user support to scrutinize (in particular for CAF/PROOF):
  - Memory requirements (leaks)
  - Correct data member initialization
    - On client and workers

Plans for scheduled analysis: Analysis train producing AODs



See talk of Andreas Morsch for detailed info

Documentation and current status of analysis modules

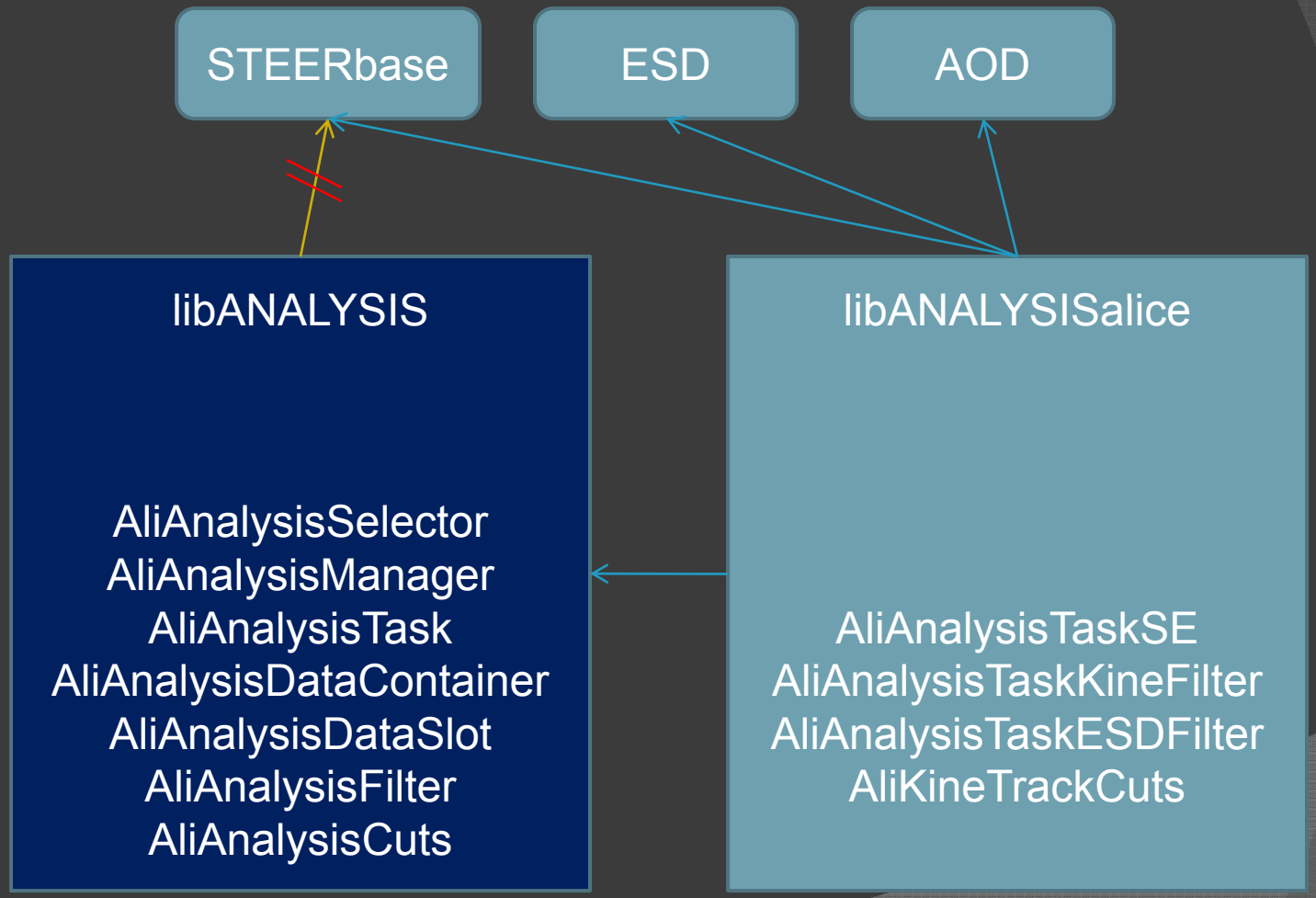
- Maintained and updated on analysis web page
  - Semantics of AliAnalysisTask methods explained with examples
- Analysis train put togeather and tested centrally on regular basis (M.Gheata)
  - As different PWG integrate their analysis code with the framework

Documentation and current status of analysis modules

NOTE: this table is not complete and will be update as soon as the developers report the correct status

Group	class	local	CAF	GRID
PWG0	AlidNdEtaTask	?	?	?
PWG0	AlidNdEtaCorrectionTask	?	?	?
PWG0	AliAnalysisTaskESDfilter	OK	OK	OK
PWG1				
PWG2	AliAnalysisTaskProtons	OK	OK	OK
PWG2	AliAnalysisTaskFemto	?	?	?
PWG2	AliAnalysisTaskESDCheckV0	?	?	?
PWG2	AliAnalysisTaskESDDedx	?	?	?
PWG2	AliAnalysisTaskESDStrange	?	?	?
PWG2	AliAnalysisTaskESDStrangeMC	?	?	?
PWG3	AliAnalysisTaskMounAODfromGeneral	?	?	?
PWG3	AliAnalysisTaskSingleMu	?	?	?
PWG3	AliAnalysisTaskVertexingHF	OK	OK	OK
PWG4	AliAnalysisTaskGamma	OK	OK	OK
PWG4	AliAnalysisTaskJets	OK	OK	OK

CODE STRUCTURE



Some future challenges

- Support for more exotic use cases
  - Mixing different inputs
  - Support more than one go over the input dataset
  - Needs support from existing infrastructures (PROOF, AliEn)
- Better support for error handling
  - Spotting where the problem is in such combined systems is not trivial
  - Improving system info inspection per task

Summary

- “Common practice” introduced by a new analysis framework developed by ALICE offline
  - Runs transparently the same user code locally, in PROOF or GRID
- Framework manages a list of independent tasks:
  - Feeding from same input data or chained one to another
  - Execution triggered by data readiness
  - Sequential execution of the top level task (train) driven by input chain (Tselector mechanism)
- Common I/O is managed by event handlers
  - ALICE-specific implementations for ESD, AOD and MC data
- Framework adopted by all PWG and well hammered on CAF and AliEn grid

Thanks to all people that contributed in one way or the other to developing, improving and testing the framework

A. Morsch
M. Gheata
J.-F. Grosse-Oetringhaus
Ch, Klein-Boesing
M. Oldenburg
F. Carminati
Y. Schutz

and many others