A Large Ion Collider Experiment

European Organisation for Nuclear Research

MAKING DISTRIBUTED ALICE ANALYSIS SIMPLE USING THE GRID PLUG-IN

Andrei Gheata[1], Mihaela Gheata[2]

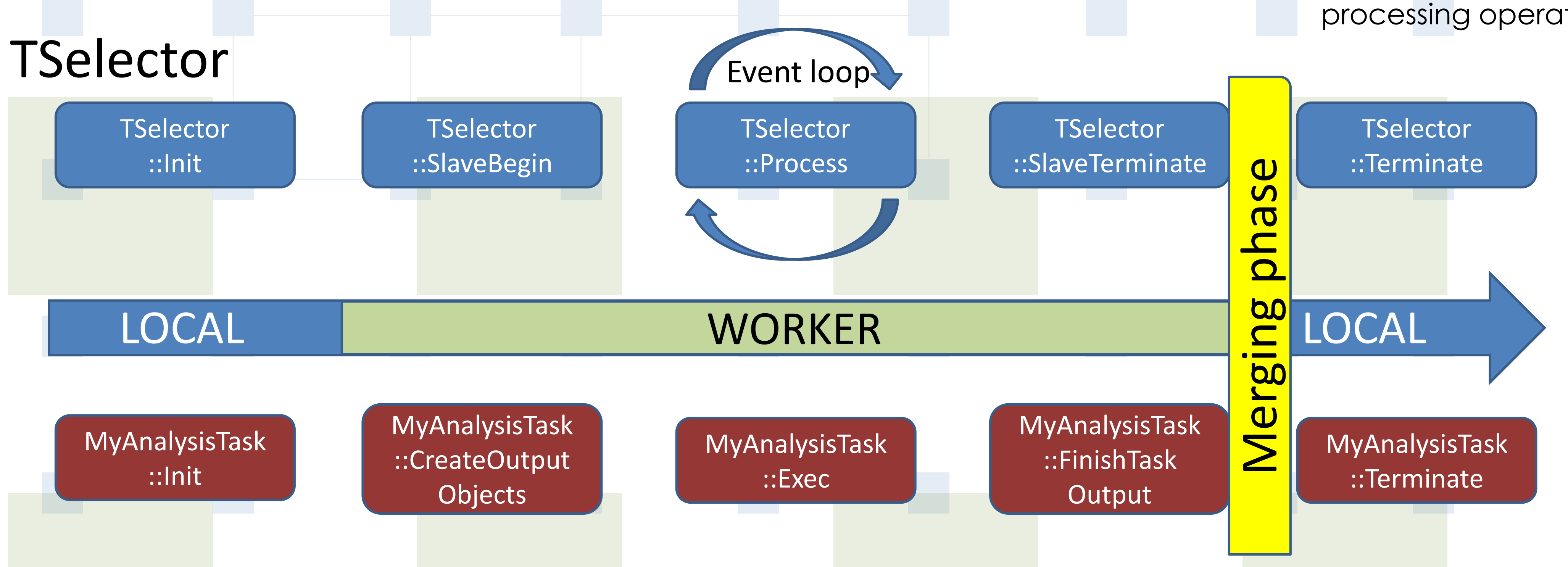
We have developed an interface within the ALICE analysis framework that allows transparent usage of the experiment's distributed resources. This analysis plug-in makes it possible to configure back-end specific parameters from a single interface and to run with no change the same custom user analysis in many computing environments, from local workstations to PROOF clusters or GRID resources. The tool is used now extensively in the ALICE collaboration for both end-user analysis and large scale productions.

Analysis modules in ALICE

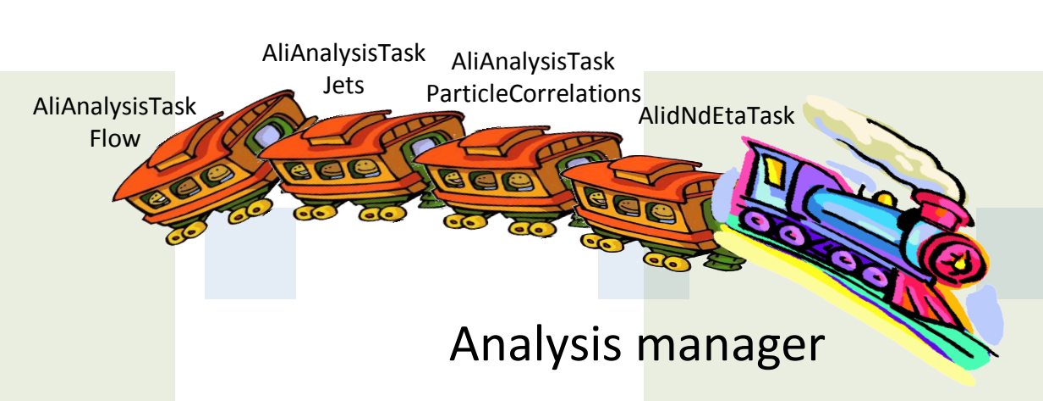
All ALICE analysis modules (aka *analysis tasks*) follow a simple formalism built on top of the ROOT TSelector:

- **Local initialization** phase (corresponding to *TSelector::Init*) - allows initialization of the custom data structures for the analysis task. This phase is typically used to configure the analysis code
- **Remote initialization** phase (corresponding to *TSelector::SlaveBegin*) – allows initializing the custom output objects produced by the analysis task (like booking histograms)
- **Event loop** user method (corresponding to *TSelector::Process*) – allows executing the custom task algorithm per event. This can be used as entry point for a specific user analysis framework
- **Post-processing** of the analysis results (corresponding to *TSelector::Terminate* – allows any post-processing operation (like normalization, fitting) on the merged results

TSelector



Analysis manager and trains



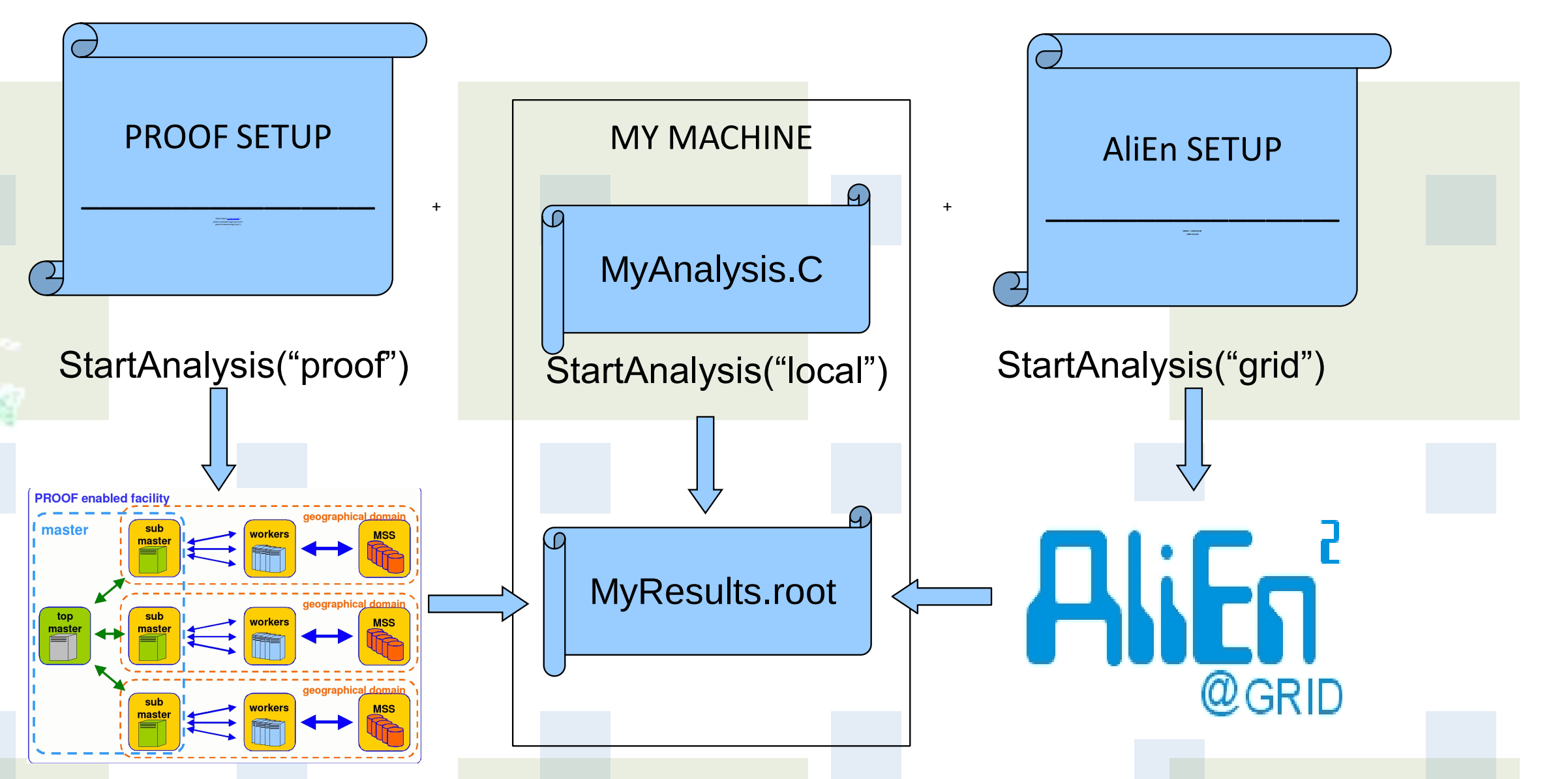
The ALICE analysis framework only requires that the user analysis follows the formalism defined by the base analysis class (*AliAnalysisTask*). All the data services, deployment and merging are provided. The analysis is steered by a single manager class that acts like a locomotive for a train of many analysis task wagons. This makes it possible to use common I/O handlers for making available for everyone the current input event and to write AOD events.

The analysis manager steers the main event loop via the TSelector mechanism. Including a new analysis to an existing train (analysis manager) is done using custom ROOT macros (so called *AddTask* macros). Once an analysis train is assembled one needs to deploy it using a specific **GRID** or **PROOF** setup. To make this operation fully transparent for the user code we developed the so called **GRID plug-in**.

Using distributed resources

ALICE uses a grid of about 85 sites that can run up to 35K jobs in parallel. The management of all these resouces is done via the **AliEn** middleware. **AliEn** is a full blast batch submission system having many features, including a general *ROOT* API called *TAlien*. This can be used to authenticate to the system and perform most commands from a ROOT session. To submit jobs using **AliEn**, one needs to define a dataset as a collection of files, a JDL file describing how the job should be run and how to collect the output, an executable and a validation script. Handling all these files can become quite difficult and error-prone, particularly when complex trains of tasks need to be handled.

The second type of distributed resources used by ALICE for fast feedback are **PROOF** clusters. These are lower scale farms that use smaller local data sets to achieve a reasonable level of interactivity in handling data quality assurance, calibration or fast data mining. The **PROOF** system is embedded in *ROOT* and comes with its own specific API and requirements that allow running analysis jobs.



http://alimonitor.cern.ch/map.jsp

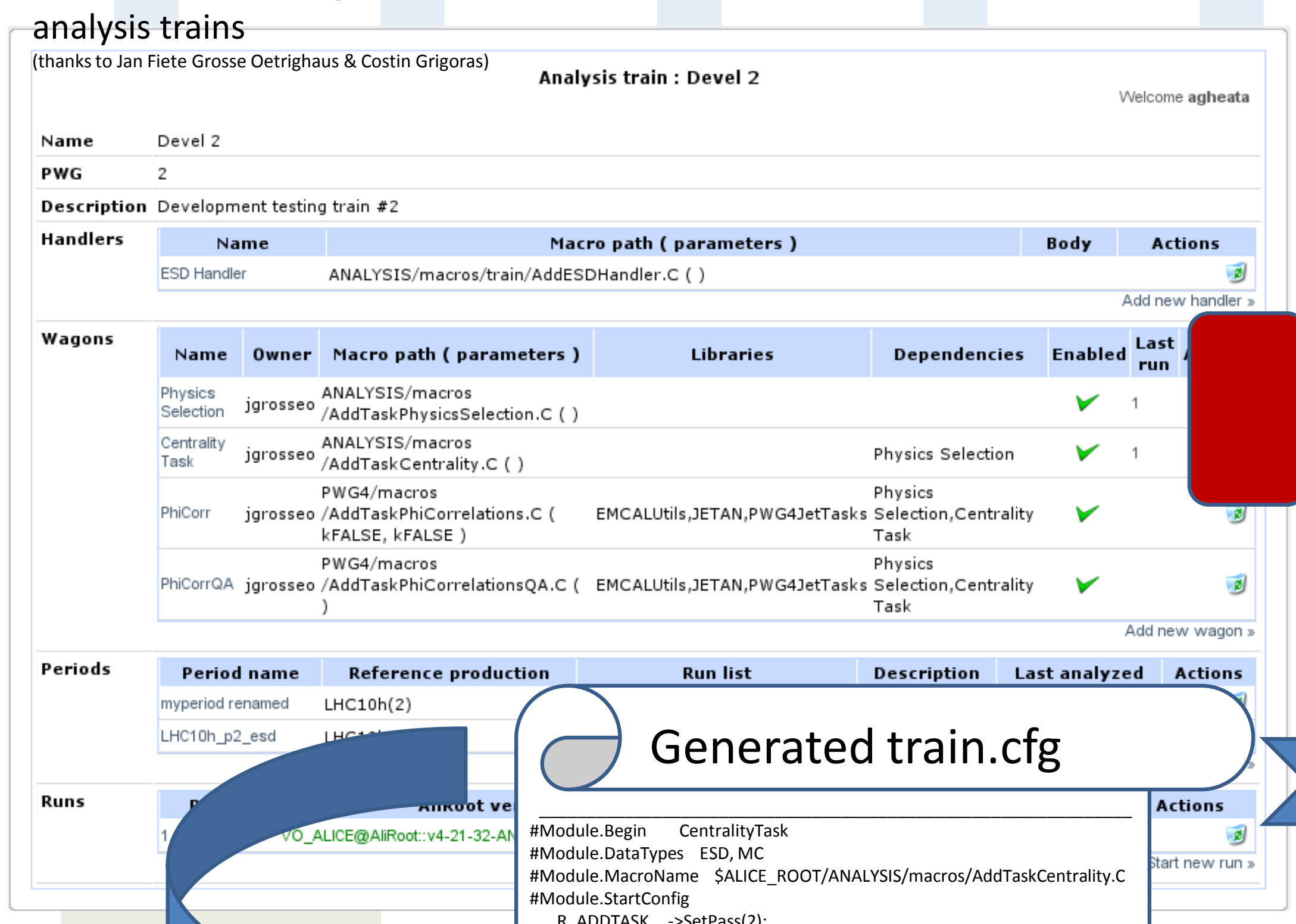
The GRID plug-in

To hide the complexity of the AliEn system from the user we have started to develop a simple utility class to describe in a more or less general way the requirements of a given analysis job. The idea was to provide a very simple API to describe in a simple key-value manner all the settings that are specific to the grid job like: time to live (TTL), software versions, data-set splitting or run numbers to be processed. Based on this settings we can automatically generate and upload the needed data-sets, JDL and analysis files. The system supports loading existing AliRoot libraries or compiling the analysis code on the fly. Using the GRID plug-in makes the analysis session look very close to a local one, the only difference coming from the few extra plug-in configuration settings.

The plug-in has been recently upgraded to handle also the local and PROOF cases, so from the user perspective the computing infrastructure becomes completely transparent. The use of this utility in ALICE is now such that it became a main component of the analysis framework. All central analysis productions are submitted based on it. Currently we are developing a web interface that will allow assembling analysis trains and generating test macros based on the analysis plug-in. This will make a lot easier the management of complex setups and will automate central productions.

New web form filling the analysis trains

(thanks to Jan Fiete Grosse Oetrighaus & Costin Grigoras)



Analysis train : Devel 2

Welcome **agheata**

Name	Devel 2
PWG	2
Description	Development testing train #2

Handlers

Name	Macro path (parameters)	Body	Actions
ESD Handler	ANALYSIS/macros/train/AddESDHandler.C ( )		

Add new handler »

Wagons

Name	Owner	Macro path (parameters)	Libraries	Dependencies	Enabled	Last run
Physics Selection	jgrosseo	ANALYSIS/macros /AddTaskPhysicsSelection.C ( )			✓	1
Centrality Task	jgrosseo	ANALYSIS/macros /AddTaskCentrality.C ( )		Physics Selection	✓	1
PhiCorr	jgrosseo	PWG4/macros /AddTaskPhiCorrelations.C (kFALSE, kFALSE)	EMCALUtils,JETAN,PWG4JetTasks	Physics Selection,Centrality Task	✓	
PhiCorrQA	jgrosseo	PWG4/macros /AddTaskPhiCorrelationsQA.C ( )	EMCALUtils,JETAN,PWG4JetTasks	Physics Selection,Centrality Task	✓	

Add new wagon »

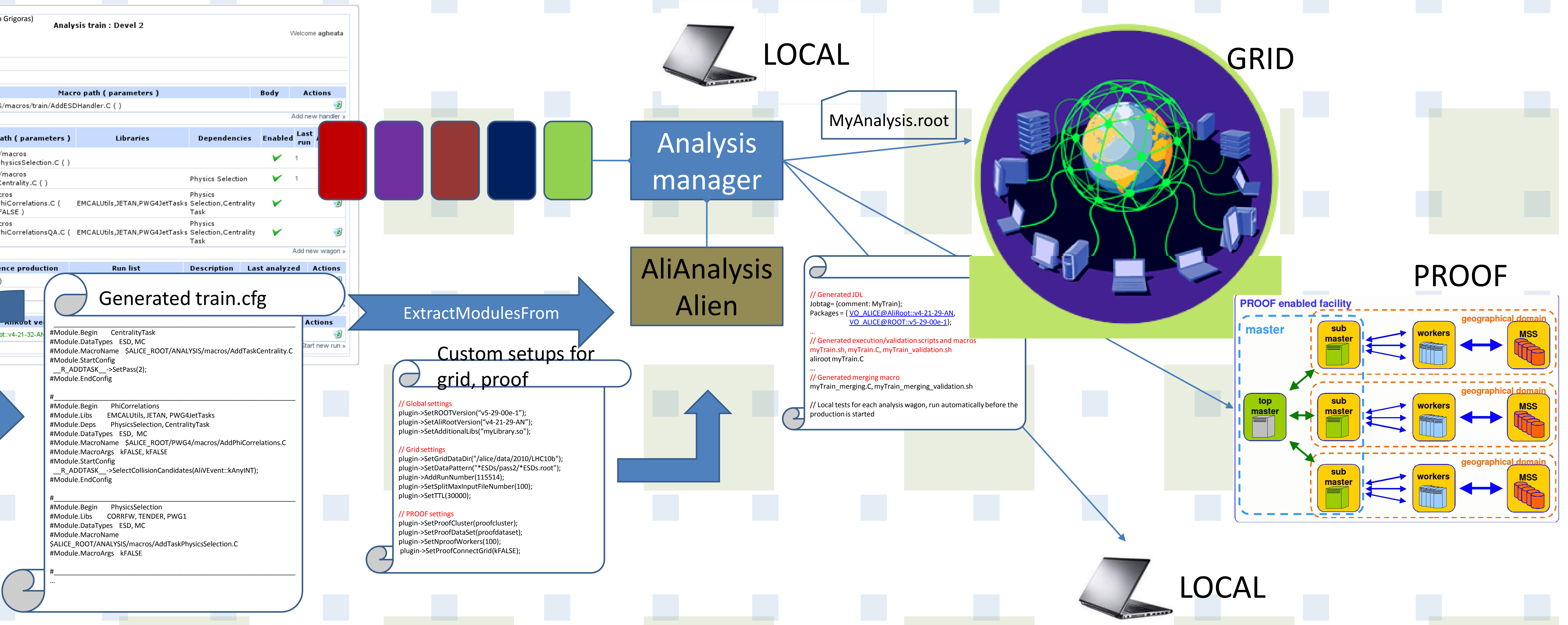
Periods

Period name	Reference production	Run list	Description	Last analyzed	Actions
myperiod renamed	LHC10h(2)				
LHC10h_p2_esd	LHC1				

Runs

	AliRoot ve		Actions
1	VO_ALICE@AliRoot::v4-21-32-AN		

Start new run »



Generated train.cfg

```
___________________________________________________
#Module.Begin      CentralityTask
#Module.DataTypes   ESD, MC
#Module.MacroName   $ALICE_ROOT/ANALYSIS/macros/AddTaskCentrality.C
#Module.StartConfig
 __R_ADDTASK__->SetPass(2);
#Module.EndConfig

#__________________________________________________
#Module.Begin      PhiCorrelations
#Module.Libs       EMCALUtils, JETAN, PWG4JetTasks
#Module.Deps       PhysicsSelection, CentralityTask
#Module.DataTypes   ESD,  MC
#Module.MacroName   $ALICE_ROOT/PWG4/macros/AddPhiCorrelations.C
#Module.MacroArgs   kFALSE, kFALSE
#Module.StartConfig
 __R_ADDTASK__->SelectCollisionCandidates(AliVEvent::kAnyINT);
#Module.EndConfig

#__________________________________________________
#Module.Begin      PhysicsSelection
#Module.Libs       CORRFW, TENDER, PWG1
#Module.DataTypes   ESD, MC
#Module.MacroName
$ALICE_ROOT/ANALYSIS/macros/AddTaskPhysicsSelection.C
#Module.MacroArgs   kFALSE

#__________________________________________________
…
```

Custom setups for grid, proof

```
// Global settings
plugin->SetROOTVersion("v5-29-00e-1");
plugin->SetAliRootVersion("v4-21-29-AN");
plugin->SetAdditionalLibs("myLibrary.so");

// Grid settings
plugin->SetGridDataDir("/alice/data/2010/LHC10b");
plugin->SetDataPattern("*ESDs/pass2/*ESDs.root");
plugin->AddRunNumber(115514);
plugin->SetSplitMaxInputFileNumber(100);
plugin->SetTTL(30000);

// PROOF settings
plugin->SetProofCluster(proofcluster);
plugin->SetProofDataSet(proofdataset);
plugin->SetNproofWorkers(100);
 plugin->SetProofConnectGrid(kFALSE);
```

```
// Generated JDL
Jobtag= {comment: MyTrain};
Packages = { VO_ALICE@AliRoot::v4-21-29-AN,
             VO_ALICE@ROOT::v5-29-00e-1};
…
// Generated execution/validation scripts and macros
myTrain.sh, myTrain.C, myTrain_validation.sh
aliroot myTrain.C
…
// Generated merging macro
myTrain_merging.C, myTrain_merging_validation.sh

// Local tests for each analysis wagon, run automatically before the production is started
```

[1]CERN, [2]Institute for Space Sciences - Bucharest