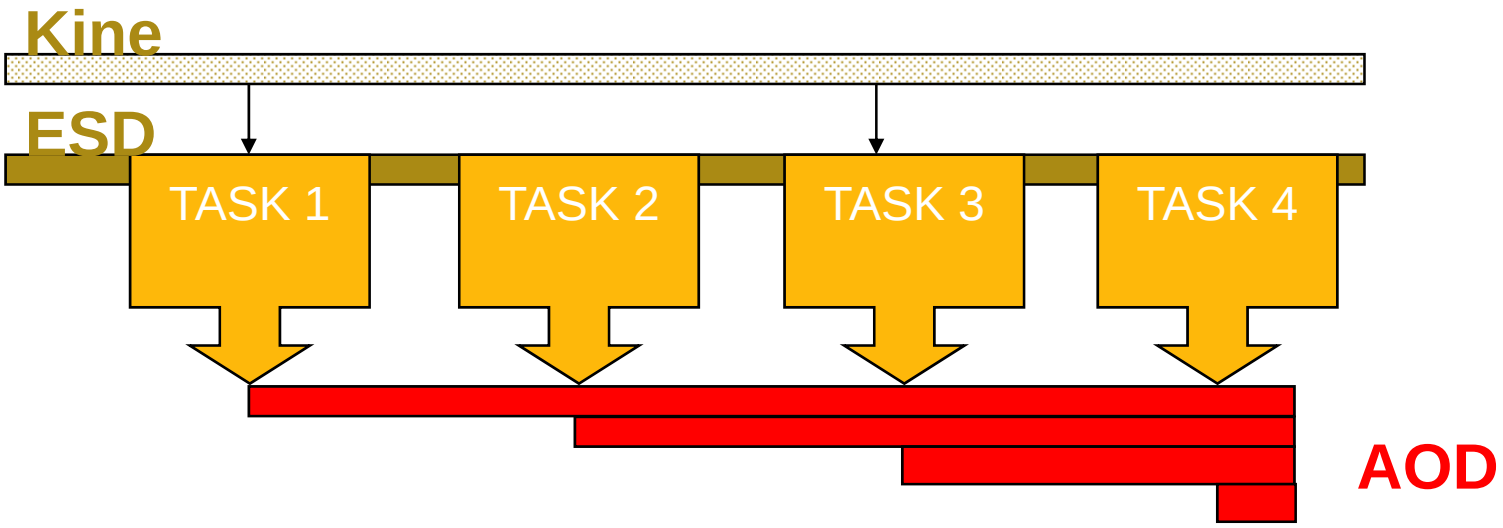
ANALYSIS TRAIN ON THE GRID

Mihaela Gheata

AOD production train



- AOD production will be organized in a ‘train’ of tasks
  - To maximize efficiency of full dataset processing
  - To optimize CPU/IO
  - Using the analysis framework
- Testing was done to see if the AOD production approach using a train of tasks adding information one-by-one really works

What was tested

- Small train (2 tasks)
  - AliAnalysisTaskESDFilter – ESD to AOD filter task
    - Filtering tracks and copying other ESD information into AOD structure
  - AliAnalysisTaskJets – task for executing jet analysis using the analysis manager framework
    - Doing jet reconstruction and writing results to AOD
- Possibility to run in different modes
  - Local (with files accessed from AliEn/CAF) – testing if the algorithm works
  - PROOF mode (CAF) – testing parallel run mode and merging
  - Grid mode – the production usage

The setup

- Many changes happened in a short time in the framework
  - Packages re-organized: base/virtual classes moved in STEERBase
  - Analysis framework: event handlers (AOD handler, MC handler)
  - Data structure (ESD,AOD) have been re-designed
- The test setup was .par based (submit code rather than use existing libs)
  - Set of base common packages: STEERBase, ESD, AOD, ANALYSIS
  - Analysis tasks: PWG0base, JETAN
  - The realistic use case will be using AliRoot libraries !

Problems on the way

- Several crashes or inconsistent results in early tests
  - Accessing old data (PDC'06) using new ESD structure was not possible for a while
  - Some fixes (protections) were added in the filtering code
  - Event loop not processing all chained files in GRID mode for tag-based analysis (uninitialized variable in TChain)
- Needed quite long to familiarize with usage of AliEn + event tag system
  - Some procedures not exactly working as described (at least for my rather exotic setup)
    - Hopefully users will be giving more and more feedback
    - More FAQ's and working examples

Splitting in GRID

- Some specific GRID issues like: how to split the input ESD chain (how many files/ sub-process)
  - Too few (e.g. 1/job) good for minimizing the failure rate but not always efficient (jobs waiting long from SPLITTING to being processed)
    - Tested 1,5,10,20,50 files/subjob (100KB to 5MB aod.root)
  - Too many – jobs starting to fail pointing to memory problems ? (or just being killed…)
    - Cannot be reproduced in local mode
    - Hard to investigate what happened
  - Efficiency depends also on processing time
    - Cannot really be tested now (filtering and jet analysis are fast)

```
# this is the startup process for root
Executable="root.sh";
Jobtag={"comment:JET analysis test"};

# we split per storage element
Split="se";

# we want each job to read 10 input files
SplitMaxInputFileNumber="10";


# this job has to run in the ANALYSIS partition
Requirements=( member(other.GridPartitions,"Analysis") );

# we need ROOT and the API service configuration package
Packages={"APISCONFIG::V2.2","VO_ALICE@ROOT::v5-15-08","VO_ALICE@AliRoot::v4-04-Rev-14"};
TTL = "30000";

#ROOT will read this collection file to know, which files to analyze
InputDataList="wn.xml";

#ROOT requires the collection file in the xml-single format
InputDataListFormat="merge:/alice/cern.ch/user/m/mgheata/test/global.xml";

# this is our collection file containing the files to be analyzed
InputDataCollection="LF:/alice/cern.ch/user/m/mgheata/test/global.xml,nodownload";

InputFile= {"LF:/alice/cern.ch/user/m/mgheata/test/runProcess.C",
            "LF:/alice/cern.ch/user/m/mgheata/test/ESD.par",
            "LF:/alice/cern.ch/user/m/mgheata/test/AOD.par",
            "LF:/alice/cern.ch/user/m/mgheata/test/ANALYSIS.par",
            "LF:/alice/cern.ch/user/m/mgheata/test/JETAN.par",
            "LF:/alice/cern.ch/user/m/mgheata/test/STEERBase.par",
            "LF:/alice/cern.ch/user/m/mgheata/test/PWG0base.par",
            "LF:/alice/cern.ch/user/m/mgheata/test/ConfigJetAnalysis.C",
            "LF:/alice/cern.ch/user/m/mgheata/test/CreateChain.C"};

# Output archive
OutputArchive={"log_archive.zip:stdout,stderr@Alice::CERN::se","root_archive.zip:*.root@Alice::CERN::se"};

# Output directory
OutputDir="/alice/cern.ch/user/m/mgheata/test/output/#alien_counter#";

# Output files
OutputFile={"aod.root","histos.root","jets_local.root"};

# Merge the output
Merge = {"aod.root:/alice/jdl/mergerootfile.jdl:aod-merged.root"};
MergeOutputDir={"/alice/cern.ch/user/m/mgheata/test/"};

# Validation
Validationcommand ="/alice/cern.ch/user/m/mgheata/bin/validate.sh";

# email
Email="Mihaela.Gheata@cern.ch";
```

AOD validation

- As well as ESD's, produced AOD's need to pass a validation test
  - Otherwise there is no control in what was really produced
    - As in AliEn's Validationcommand={…}
- I implemented this in a very simple way
  - Trying to open aod.root and writing a tiny file in case of success
    - No need be too elaborated at this level
      - Can be refined per run (doing QA histograms…)

```
# this is the startup process for root
Executable="root.sh";
Jobtag={"comment:JET analysis test"};

# we split per storage element
Split="se";

# we want each job to read 10 input files
SplitMaxInputFileNumber="10";


# this job has to run in the ANALYSIS partition
Requirements=( member(other.GridPartitions,"Analysis") );

# we need ROOT and the API service configuration package
Packages={"APISCONFIG::V2.2","VO_ALICE@ROOT::v5-15-08","VO_ALICE@AliRoot::v4-04-Rev-14"};
TTL = "30000";

#ROOT will read this collection file to know, which files to analyze
InputDataList="wn.xml";

#ROOT requires the collection file in the xml-single format
InputDataListFormat="merge:/alice/cern.ch/user/m/mgheata/test/global.xml";

# this is our collection file containing the files to be analyzed
InputDataCollection="LF:/alice/cern.ch/user/m/mgheata/test/global.xml,nodownload";

InputFile= {"LF:/alice/cern.ch/user/m/mgheata/test/runProcess.C",
        "LF:/alice/cern.ch/user/m/mgheata/test/ESD.par",
        "LF:/alice/cern.ch/user/m/mgheata/test/AOD.par",
        "LF:/alice/cern.ch/user/m/mgheata/test/ANALYSIS.par",
        "LF:/alice/cern.ch/user/m/mgheata/test/JETAN.par",
        "LF:/alice/cern.ch/user/m/mgheata/test/STEERBase.par",
        "LF:/alice/cern.ch/user/m/mgheata/test/PWG0base.par",
        "LF:/alice/cern.ch/user/m/mgheata/test/ConfigJetAnalysis.C",
        "LF:/alice/cern.ch/user/m/mgheata/test/CreateChain.C"};

# Output archive
OutputArchive={"log_archive.zip:stdout,stderr@Alice::CERN::se","root_archive.zip:*.root@Alice::CERN::se"};

# Output directory
OutputDir="/alice/cern.ch/user/m/mgheata/test/output/#alien_counter#";

# Output files
OutputFile={"aod.root","histos.root","jets_local.root"};

# Merge the output
Merge = {"aod.root:/alice/jdl/mergerootfile.jdl:aod-merged.root"};
MergeOutputDir={"/alice/cern.ch/user/m/mgheata/test/"};

# Validation
Validationcommand ="/alice/cern.ch/user/m/mgheata/bin/validate.sh";

# email
Email="Mihaela.Gheata@cern.ch";
```

Merging AOD's

- We may want have some strategy of storing AOD's for a run
- CAF merging OK
  - Using wrappers from AliAnalysisManager
- A problem that scales with the size of the train
  - Filtering + jets ~ 100MB/run (100K pp events)
  - Merging has to fit memory
- More parameters in the automatic merging in AliEn ?
  - To follow the desired granularity depending of the number of job input files and of the AOD size
  - Problems with TFileMerger
    - If non-mergeable object in file, trees not merged (extra TProcessId object – put by whom if files were validated?)

```
# this is the startup process for root
Executable="root.sh";
Jobtag={"comment:JET analysis test"};

# we split per storage element
Split="se";

# we want each job to read 10 input files
SplitMaxInputFileNumber="10";


# this job has to run in the ANALYSIS partition
Requirements=( member(other.GridPartitions,"Analysis") );

# we need ROOT and the API service configuration package
Packages={"APISCONFIG::V2.2","VO_ALICE@ROOT::v5-15-08","VO_ALICE@AliRoot::v4-04-Rev-14"};
TTL = "30000";

#ROOT will read this collection file to know, which files to analyze
InputDataList="wn.xml";

#ROOT requires the collection file in the xml-single format
InputDataListFormat="merge:/alice/cern.ch/user/m/mgheata/test/global.xml";

# this is our collection file containing the files to be analyzed
InputDataCollection="LF:/alice/cern.ch/user/m/mgheata/test/global.xml,nodownload";

InputFile= {"LF:/alice/cern.ch/user/m/mgheata/test/runProcess.C",
        "LF:/alice/cern.ch/user/m/mgheata/test/ESD.par",
        "LF:/alice/cern.ch/user/m/mgheata/test/AOD.par",
        "LF:/alice/cern.ch/user/m/mgheata/test/ANALYSIS.par",
        "LF:/alice/cern.ch/user/m/mgheata/test/JETAN.par",
        "LF:/alice/cern.ch/user/m/mgheata/test/STEERBase.par",
        "LF:/alice/cern.ch/user/m/mgheata/test/PWG0base.par",
        "LF:/alice/cern.ch/user/m/mgheata/test/ConfigJetAnalysis.C",
        "LF:/alice/cern.ch/user/m/mgheata/test/CreateChain.C"};

# Output archive
OutputArchive={"log_archive.zip:stdout,stderr@Alice::CERN::se","root_archive.zip:*.root@Alice::CERN::se"};

# Output directory
OutputDir="/alice/cern.ch/user/m/mgheata/test/output/#alien_counter#";

# Output files
OutputFile={"aod.root","histos.root","jets_local.root"};

# Merge the output
Merge = {"aod.root:/alice/jdl/mergerootfile.jdl:aod-merged.root"};
MergeOutputDir={"/alice/cern.ch/user/m/mgheata/test/"};

# Validation
Validationcommand ="/alice/cern.ch/user/m/mgheata/bin/validate.sh";

# email
Email="Mihaela.Gheata@cern.ch";
```

What we have learned

- The system is complex enough…
  - Testing all changes at a time does not help much in debugging it
- … but it works
  - The analysis framework stable in all modes
    - CAF case working from the beginning
  - AOD production can be done following a rather simple pattern
  - Testing in local mode with files from GRID was very useful for debugging

Overview and conclusions

- Tested a small AOD train based on AliAnalysisManager framework
  - Several problems came up but the global result OK
  - Strategy related to job splitting, output merging and validation to be established
  - Train size may influence **.jdl** parameters
- Analysis framework running stable in all modes
  - Running successfully in local mode is a good hint for what will happen in PROOF or GRID
  - Using such trains may lead to memory issues that should be understood before running in GRID