Analysis with PROOF in ALICE

Arsen Hayrapetyan, Yerevan Physics Institute; CERN
Arsen.Hayrapetyan@cern.ch

Outline

Part 1: Theory

- PROOF, AAF, CAF
- PROOF Analysis. Merging, submerging
- PROOF terminology
- The structure of the PROOF analysis task
- Analysis data: Trees, Chains, Datasets.
- AliROOT usage options

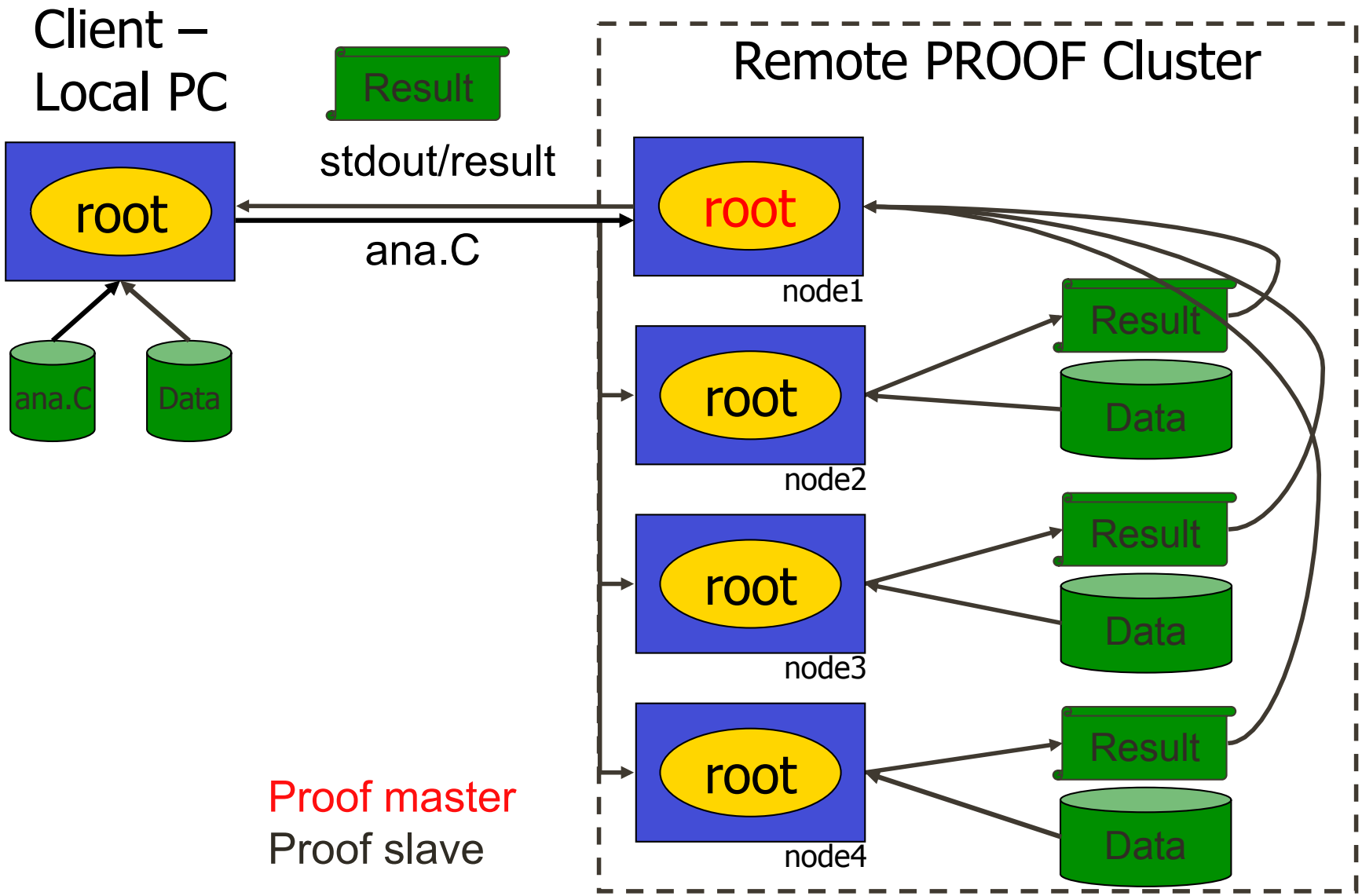
Part 2: Practice

- Exercise 0: Connecting to CAF, listing anlaysis packages and data
- Exercise 1: ESD analysis on real data on CAF
- Exercise 2: ESD analysis on MC data on CAF
- Exercise 3: Combining exercises 1 and 2
- Exercise 4: AOD analysis on real data on CAF
- Exercise 5: Staging datasets on CAF
- Exercise 6: Processing staged datasets on CAF

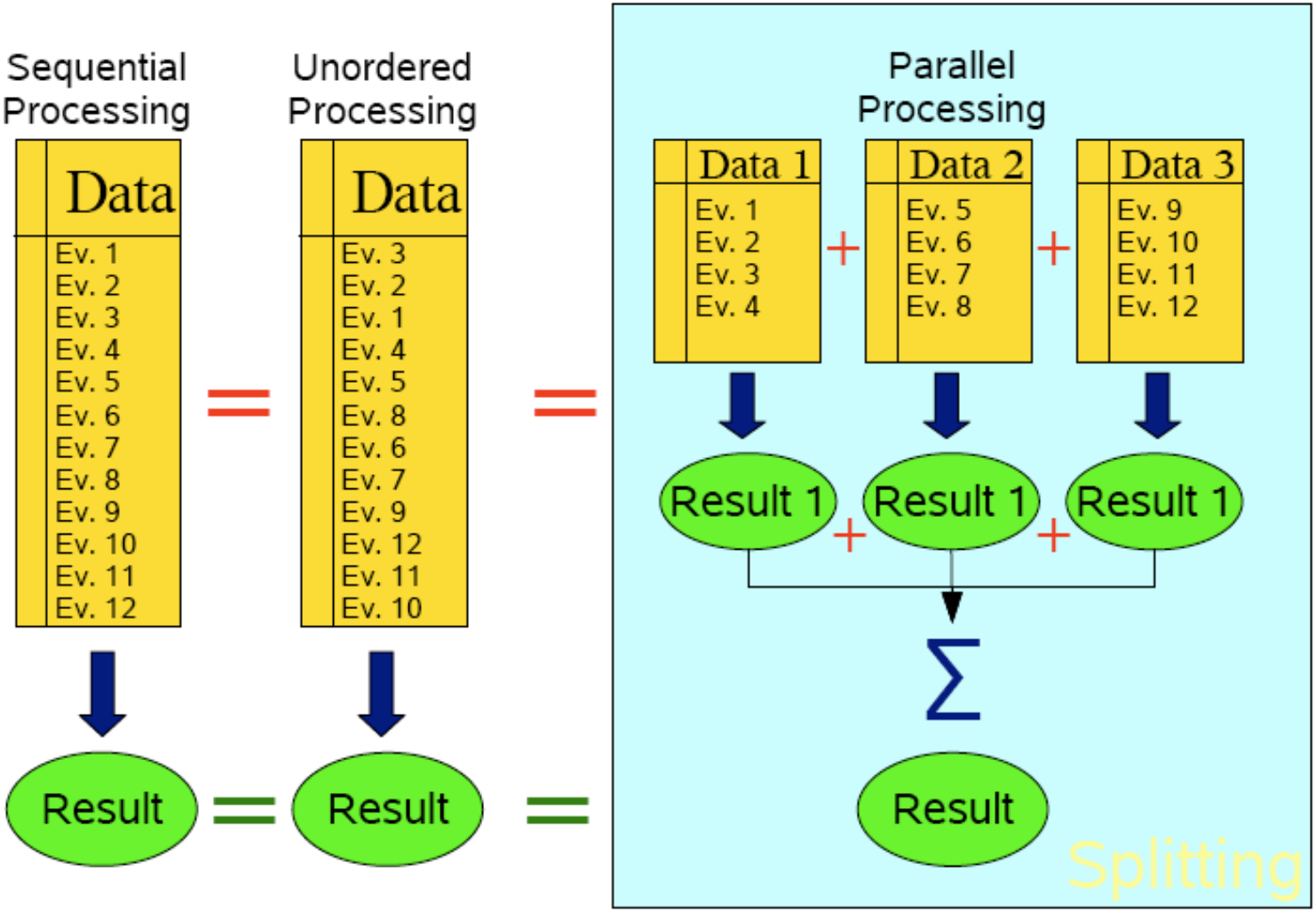
PROOF, AAF, CAF

- **P**arallel **ROO**t **F**acility is an extension of ROOT for interactive analysis of large sets of ROOT files in parallel on clusters of computers (analysis facility) or multi-core machines (PROOF Lite).
- **A**LICE **A**nalysis **F**acilities is a group of analysis facilities dedicated to prompt analysis of relatively small (compared to grid) amount of *pp* and *PbPb* data (all AFs) as well as for reconstruction of samples of raw data during data taking (CAF).
- **C**ERN **A**nalysis **F**acility is a PROOF cluster with **464 CPU** cores, **1.4 TB** total RAM and **162 TB** total disk space. The analysis data is normally staged (copied from grid) to CAF by users or administrator, before analysing it.

PROOF analysis schema



Event based parallelism



Merging of the results

- **Option 1**: Merging on the *Master*
  - The results produced on *Workers* are all sent to the *Master* and merged there
- **Option 2**: Submerging
  - Certain nodes are selected to be submergers, the results produced on *Workers* are divided between them and merged (producing smaller output), the outputs are sent to the *Master* to be finally merged
    - The number of submergers can be chosen automatically (default proof behaviour) or specified by the user

- Standard merging implementation for histograms is available
- Other classes need to implement ***Merge(TCollection)***
- When no merging function is available, individual objects are returned

PROOF terminology

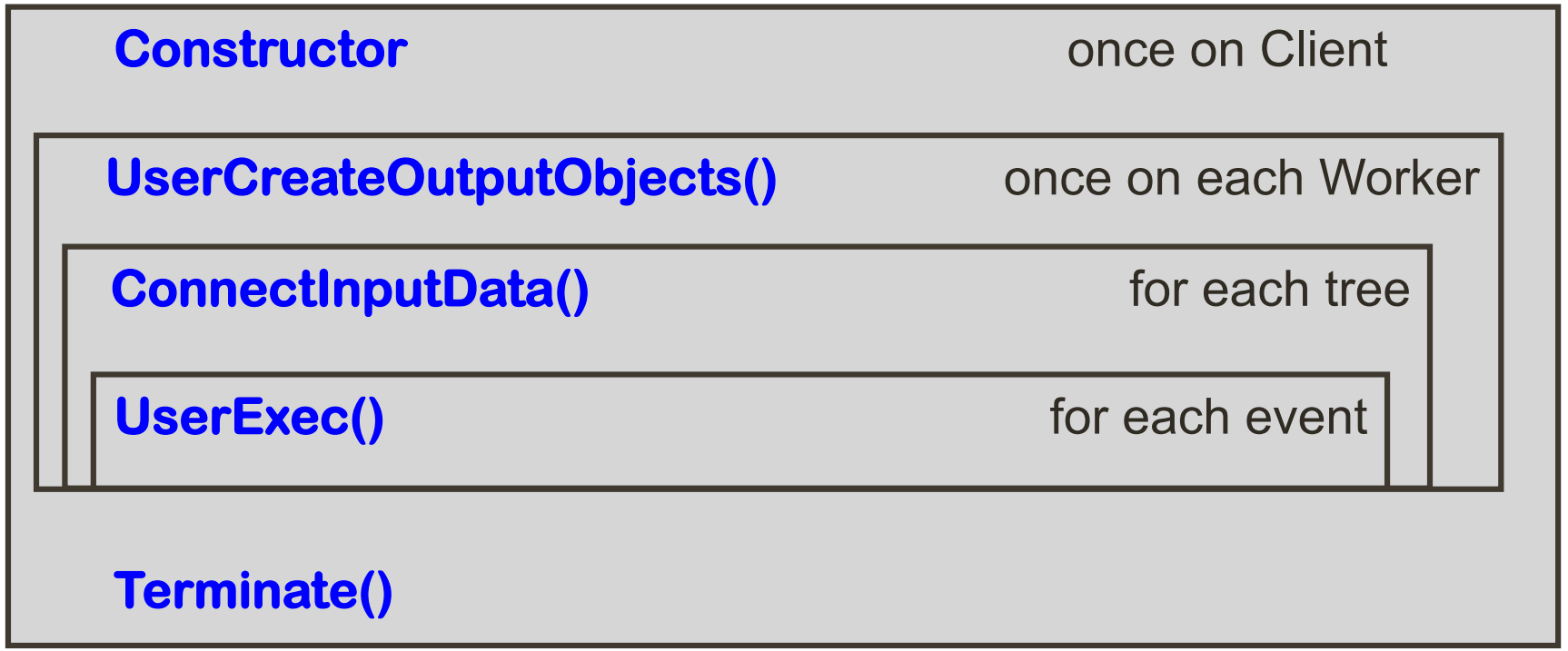
- PROOF cluster
  - A set of computers working in coordinate way following PROOF protocol
- Node
  - Single computer in PROOF cluster
- Client
  - A process created within ROOT session on your machine that is connected to Master
- Master
  - A process on a dedicated node coordinating work between workers
- Worker or Slave
  - A process on a node that processes data and is connected to the Master
- Query
  - A job submitted from the Client to Master. The query consists of a selector and a chain
- Selector
  - A class containing the analysis code
    - In ALICE we use the Analysis Framework and the Selector is usually derived from ***AliAnalysisTaskSE***
- Chain
  - A list of files (trees) to process

How does AAF analysis work

- In ALICE, we use Analysis Framework to write PROOF-enabled programs
  - Analysis task is written as a class derived from ***AliAnalysisTaskSE***
  - The data to be analysed is normally staged on AAF (if not you can ask to stage it or do it yourself). The dataset name is then specified in the steering macro
    - In case you work with PROOF Lite you can put the files containing the data into a *chain* and specify the TChain object in the steering macro
  - If you need libraries not contained in AliROOT you should pack them in PAR (PROOF Archive) packages, upload them to AAF and enable them in AAF in the steering macro

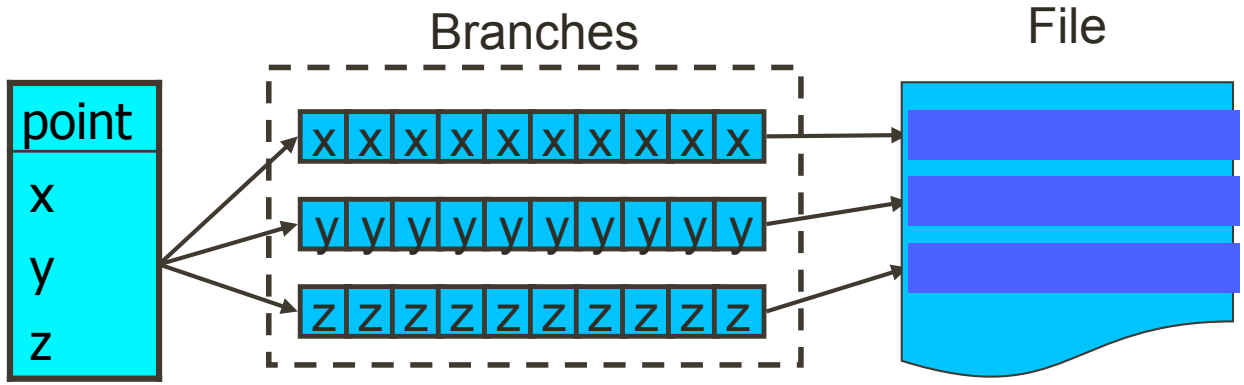
The structure of the PROOF analysis task

The (ALICE-specific) analysis task is defined via class derived from AliAnalisysTaskSE.



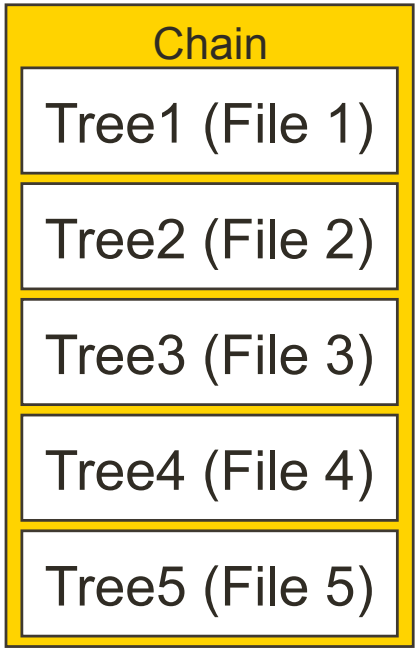
Trees

- The tree (object of ROOT class **TTree**) is a container for data storage
  - Consists of several branches
    - Can be stored in one or several files
    - Stored contiguously
    - Can be switched off during data reading (hence speed-up)
  - Content visualisation with helper functions: ***Draw()***, ***Scan()***
  - Compressed



Chains

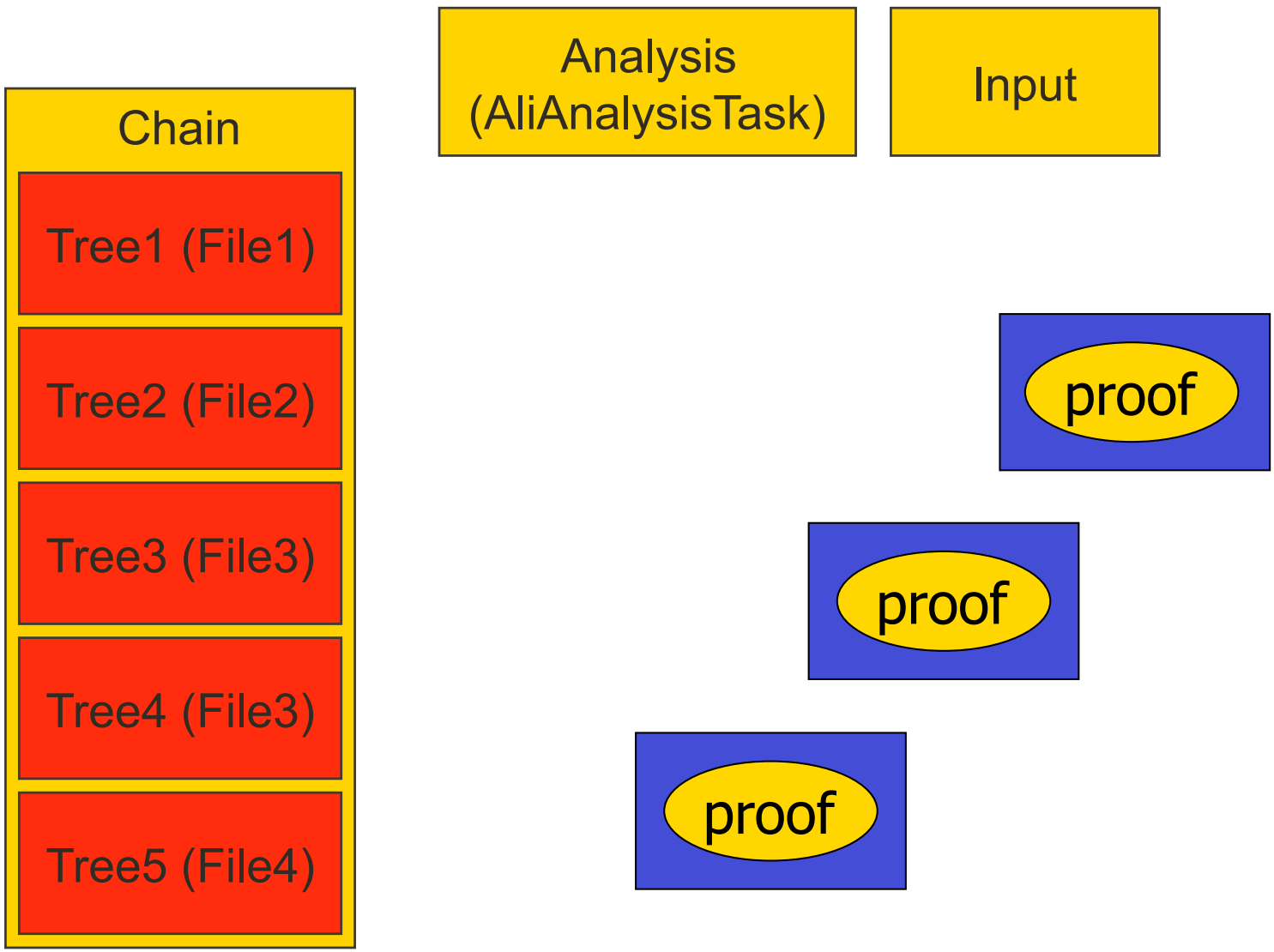
- The chain (object of ROOT class TChain) is a collection of files containing trees (TTree objects)
  - Visualisation functions ***Draw()*** and ***Scan()*** can be used, as with trees (they will iterate over all elements of the chain)
- The data to be analysed is normally put in a *tree* or *chain* for local analysis. For analysis on a PROOF cluster one uses datasets.



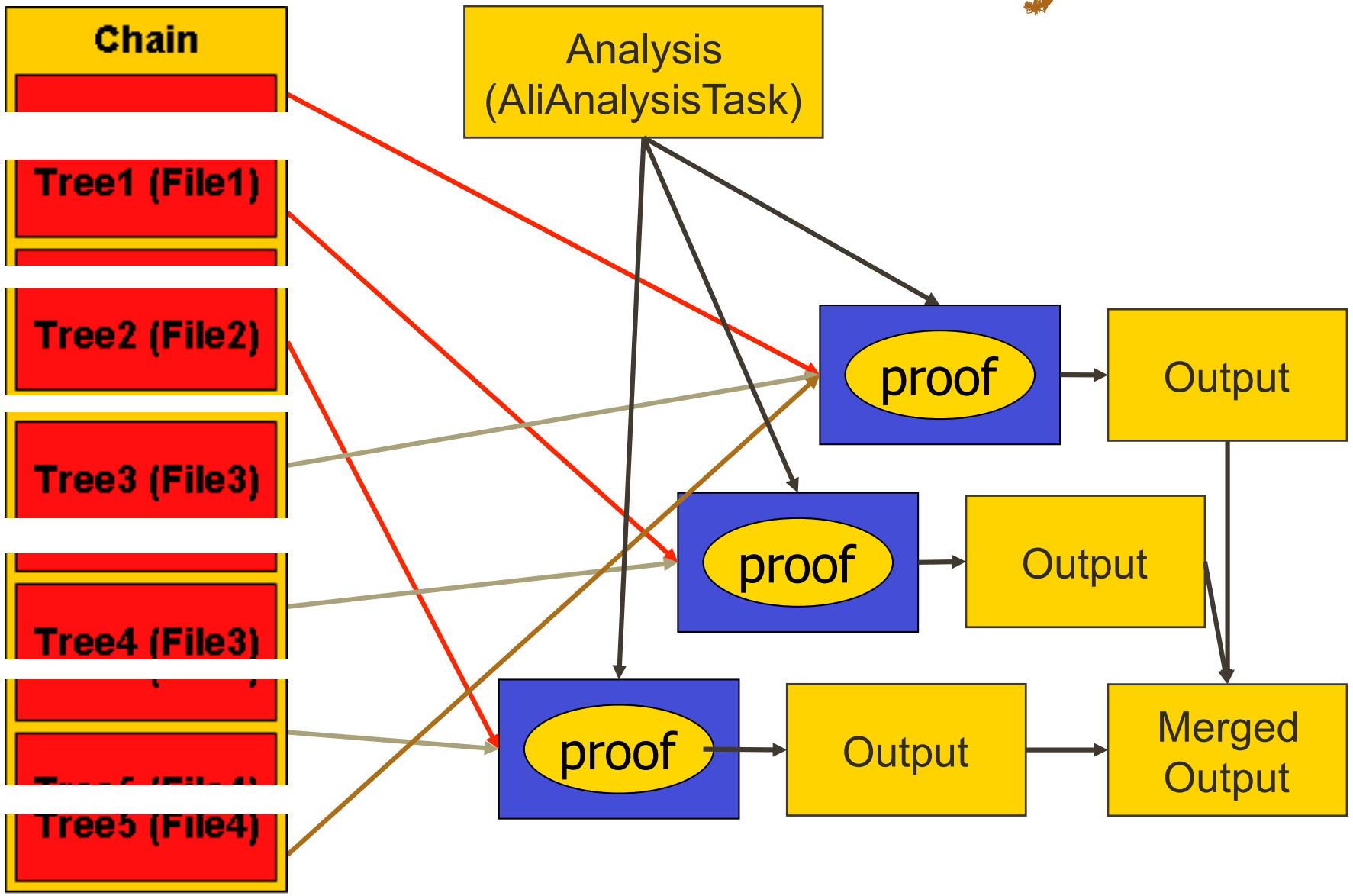
Datasets

- The dataset is a named list of files (containing trees in case of ALICE data) including metadata information about files' locations.
  - Staged to AAF by cluster administrators or users.
    - If staged by administrator, have the names starting with ***/alice/data*** or ***/alice/sim***, e.g.:
      - **/alice/data/LHC10d_000126285_p2**
      - **/alice/sim/LHC10e13_118507**
    - If staged by user, have the names starting with ***<user_group>/ <user_grid_login_name>***, e.g.:
      - **/PWG4/esicking/LHC10e20_130840_AOD060**
      - **/VZERO/cheynis/run136104_pass1**
      - Users who do not enter in any PWG or detector group, have <user_group>=default, e.g.:
        - **/default/poghos/LHC11a_000146746_pass3_with_SDD**
  - Can be listed with ***TProof::ShowDataSets()*** or via http://aaf.cern.ch -> Favourite links => AAF datasets

Workflow summary



Workflow summary



AliROOT usage options

- As main analysis software package AliROOT should be loaded into memory of the Workers on AAF before processing the data. This is done via method ***TProof::EnablePackage()***, e.g.:
  - **gProof->EnablePackage("VO_ALICE@AliRoot::v5-03-21-AN")**
  - TProof::EnablePackage() accepts one of pre-defined string constants as second parameter:
    - "default" – loads basic analysis libraries (libVMC, libTree, libPhysics, libMatrix, libMinuit, libXMLParser, LibGui, libSTEERBase, libESD, libAOD, libANALYSIS, libOADB, alibANALYSISalice), e.g.:
      - **gProof->EnablePackage("VO_ALICE@AliRoot::v5-03-21-AN", "default")**
    - "ALIROOT" – same as "default", loads libraries defined in $ALICE_ROOT/macros/loadlibs.C
      - **gProof->EnablePackage("VO_ALICE@AliRoot::v5-03-21-AN", "ALIROOT")**
    - "REC" – suited for reconstruction, loads libraries defined in $ALICE_ROOT/macros/loadlibsrec.C
    - **gProof->EnablePackage("VO_ALICE@AliRoot::v5-03-21-AN", "REC")**
    - "SIM" – suited for simulation, loads libraries defined in $ALICE_ROOT/macros/loadlibssim.C
      - **gProof->EnablePackage("VO_ALICE@AliRoot::v5-03-21-AN", "SIM")**
- The list of available packages can be displayed with ***TProof::ShowPackages()***, e.g.:
  - **gProof->ShowPackages()**

More information

- **AAF user documentation**
  - http://aaf.cern.ch/node/89
- **PROOF documentation**
  - http://root.cern.ch/drupal/content/proof
- **Analysis framework documentation**
  - http://aliweb.cern.ch/Offline/Activities/Analysis/AnalysisFramework/index.html

Part 2: Hands-on exercises

- Set up your credentials
  - $> mkdir .globus
    - Put your certificate and private key there
- Download tutorial files from agenda and unpack them

Exercise 0: Connecting to CAF, listing anlaysis packages and data

- $> root -l
- root > gEnv->SetValue("XSec.GSI.DelegProxy", "2”)
- root> TProof::Open(ahairape@alice-caf.cern.ch)
- root> gProof->ShowPackages()
- root> gProof->ShowDataSets()

Exercise 1: ESD analysis on real data on CAF

- $> cd ex1
- Inspect the files (steering macro and analysis task)
- Run the analysis
  - root -l ex1.cxx

Exercise 2: ESD analysis on MC data on CAF

- $> cd ex2
- Inspect the files (steering macro and analysis task)
- Run the analysis
  - root -l ex2.cxx

Exercise 3: Combining exercises 1 and 2

- $> cd ex3
- Inspect the files (steering macro and analysis task)
- Run the analysis
  - root -l ex3.cxx

Exercise 4: AOD analysis on real data on CAF

- $> cd ex4
- Inspect the files (steering macro and analysis task)
- Run the analysis
  - root -l ex4.cxx

Exercise 5: Staging datasets

- **Reference: http://aaf.cern.ch/node/224**
- mkdir ex5 && cd ex5
- wget http://afdsmgrd.googlecode.com/svn/tags/v1.0.6/macros/CreateDataSetFromAliEn.C

- **Task: Edit the file CreateDataSetFromAliEn.C to stage a dataset containing at most 10 files.**
  - Use ESD pass2 data for run #188359 (alien path: /alice/data/2012/LHC12g/000188359/ESDs/pass2)
  - Stage root_archive.zip files, use "AliESDs.root" as anchor
  - Specify "/esdTree" for the tree name
  - Name the dataset "testDS"
  - Test your modifications with "root -l CreateDataSetFromAliEn.C" to make sure the dataset satisfies the conditions above.
    - Once it is OK, enable actual staging with "commit" option
  - Monitor the staging progress with gProof->ShowDataSets()
    - Solution available on agenda page

Exercise 6: Analysing staged dataset

- $> cd ex1

- **Task: Modify the ex1.cxx file to analyse the dataset you have staged for Exercise 5.**