Part III: PROOF

Jan Fiete Grosse-Oetringhaus – CERN

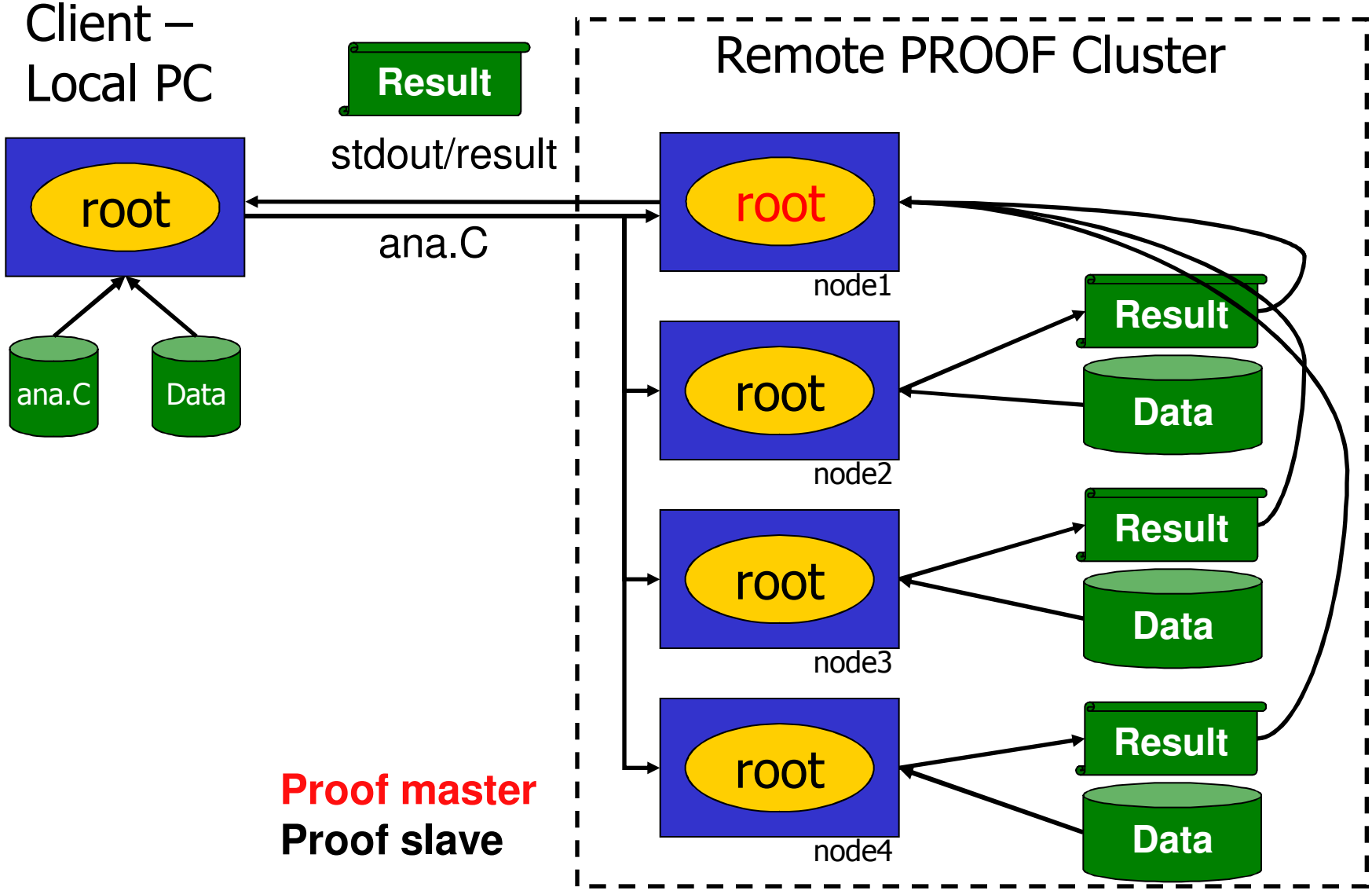
Andrei Gheata - CERN

V3.2 – 04.03.10

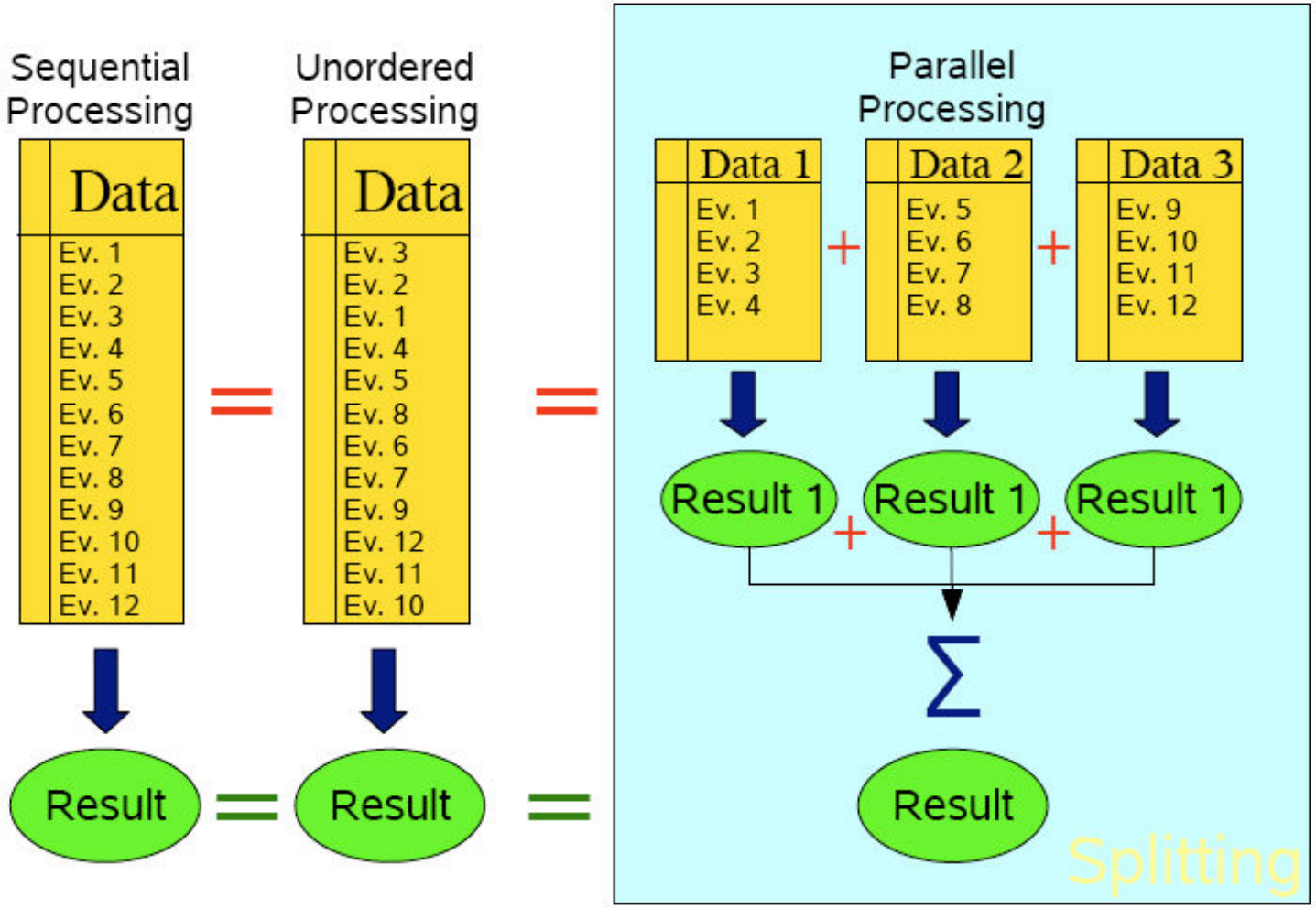
PROOF

- Parallel ROOT Facility
- Interactive parallel analysis on a local cluster
  - Parallel processing of (local) data (trivial parallelism)
  - Fast Feedback
  - Output handling with direct visualization
  - **Not** a batch system
- PROOF itself is not related to Grid
  - Can access Grid files
- The usage of PROOF is transparent
  - The same code can be run locally and in a PROOF system (certain rules have to be followed)
- PROOF is part of ROOT

PROOF Schema



Event based (trivial) Parallelism



Terminology

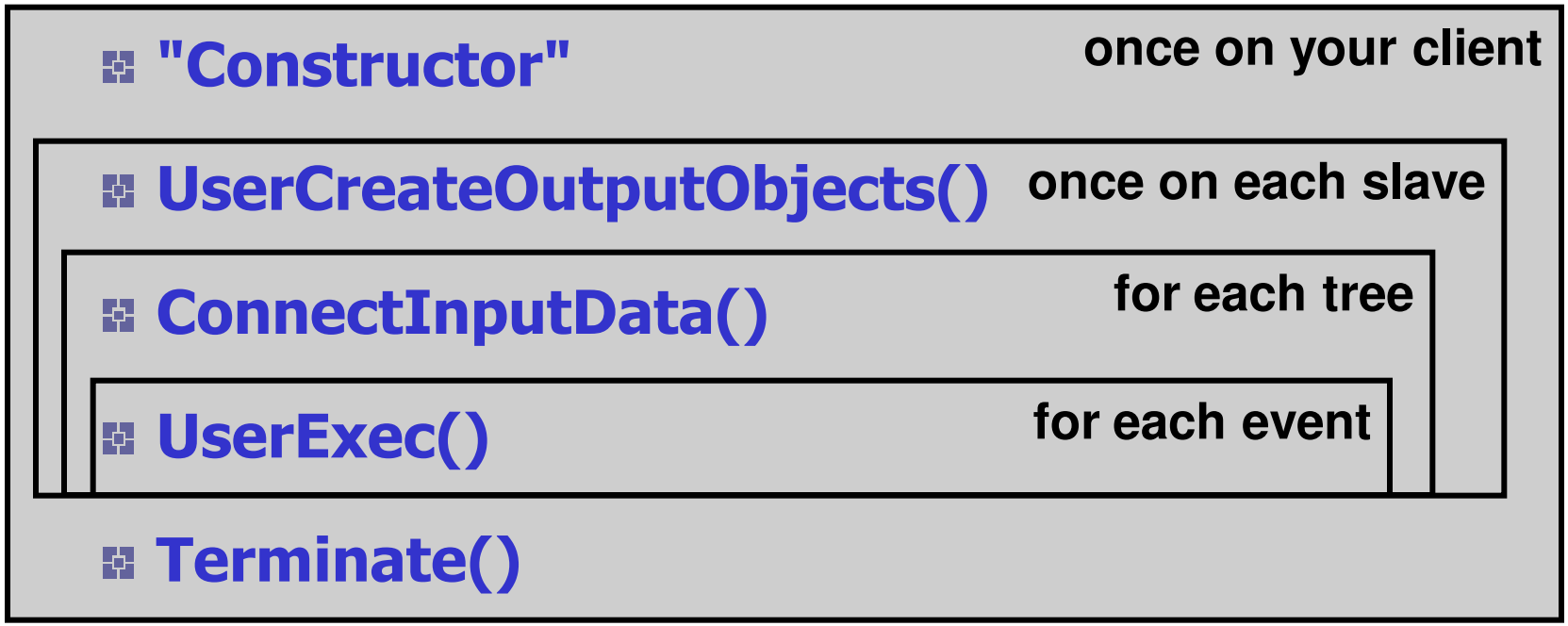
- Client
  - Your machine running a ROOT session that is connected to a PROOF master
- Master
  - PROOF machine coordinating work between slaves
- Slave/Worker
  - PROOF machine that processes data
- Query
  - A job submitted from the client to the PROOF system. A query consists of a selector and a chain
- Selector
  - A class containing the analysis code
  - In ALICE we use the Analysis Framework, therefore a AliAnalysisTaskSE is sufficient
- Chain
  - A list of files (trees) to process (more details later)

How to use PROOF

- The analysis framework is used
  - Files to be analyzed are put into a chain → TChain
  - Analysis written as a task (already introduced in previous tutorial) → AliAnalysisTaskSE
  - The same analysis like written previously can be used
- If additional libraries are needed, these have to be distributed as a "package"

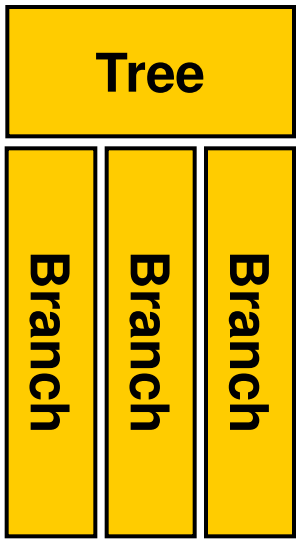
AliAnalysisTaskSE

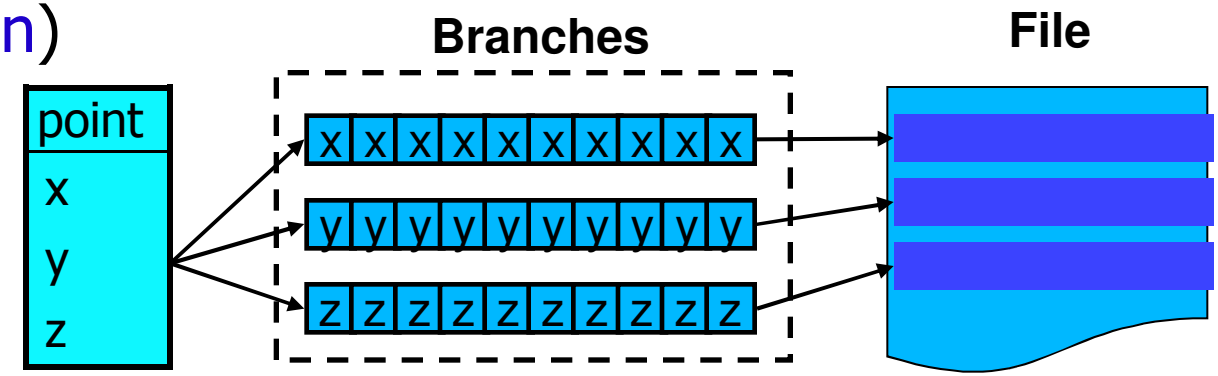
- Classes derived from AliAnalysisTaskSE can run locally, in PROOF and in AliEn



Class TTree

- A tree is a container for data storage
- It consists of several *branches*
  - These can be in one or several files
  - Branches are stored contiguously (split mode)
  - When reading a tree, certain branches can be switched off → speed up of analysis when not all data is needed
- Set of helper functions to visualize content (e.g. Draw, Scan)
- Compressed

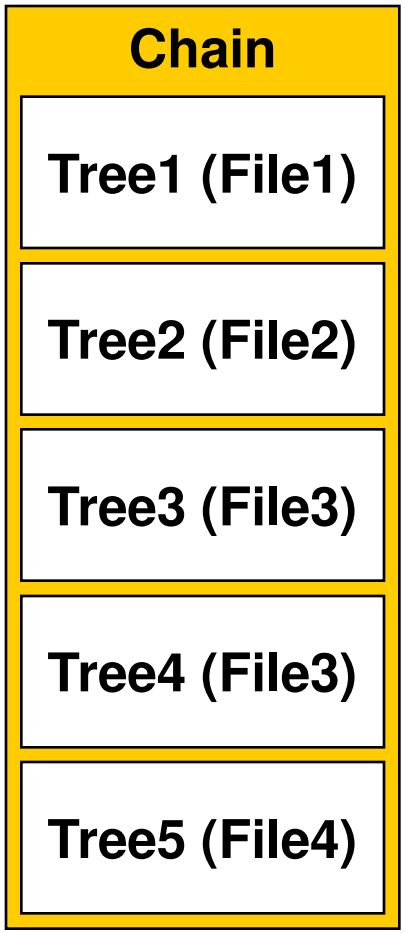




TChain

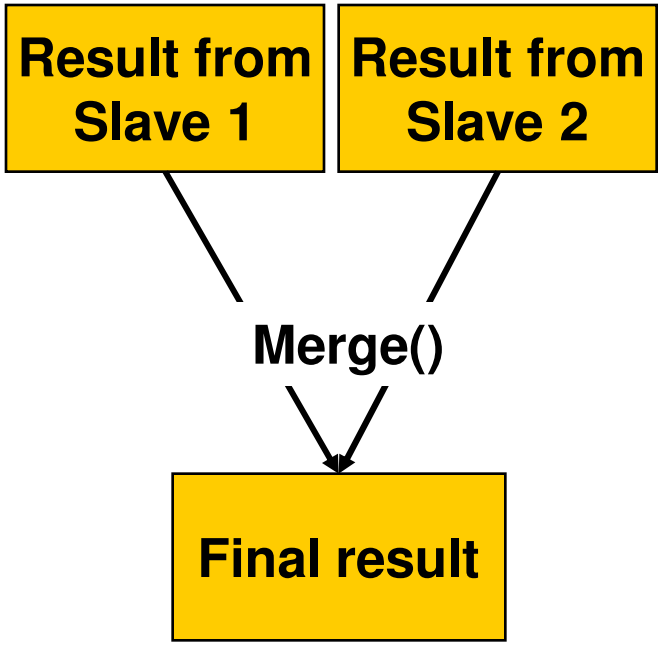
- A chain is a list of trees (in several files)
- Normal TTree functions can be used
  - **Draw(...), Scan(...)**

  → these iterate over all elements of the chain

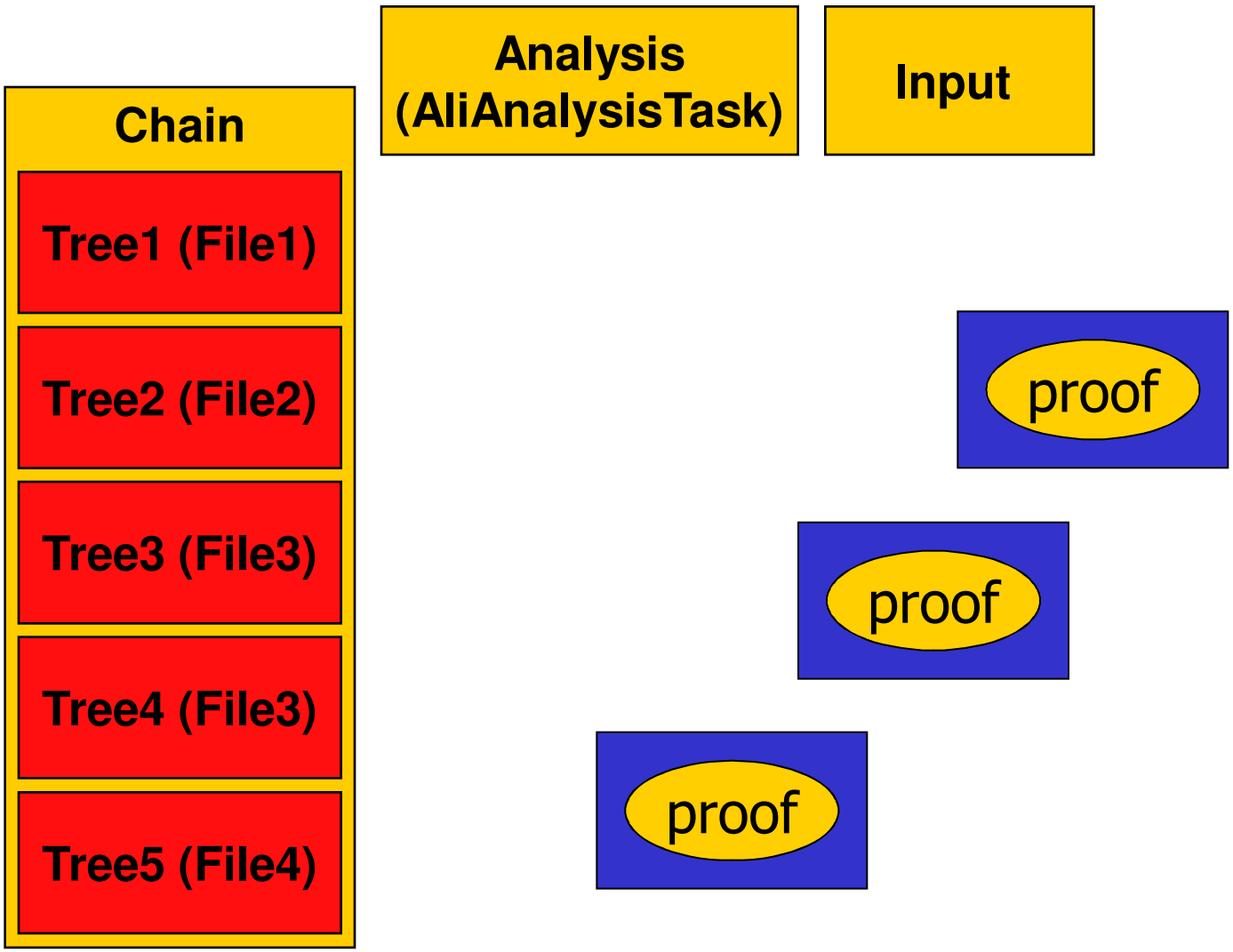


Merging

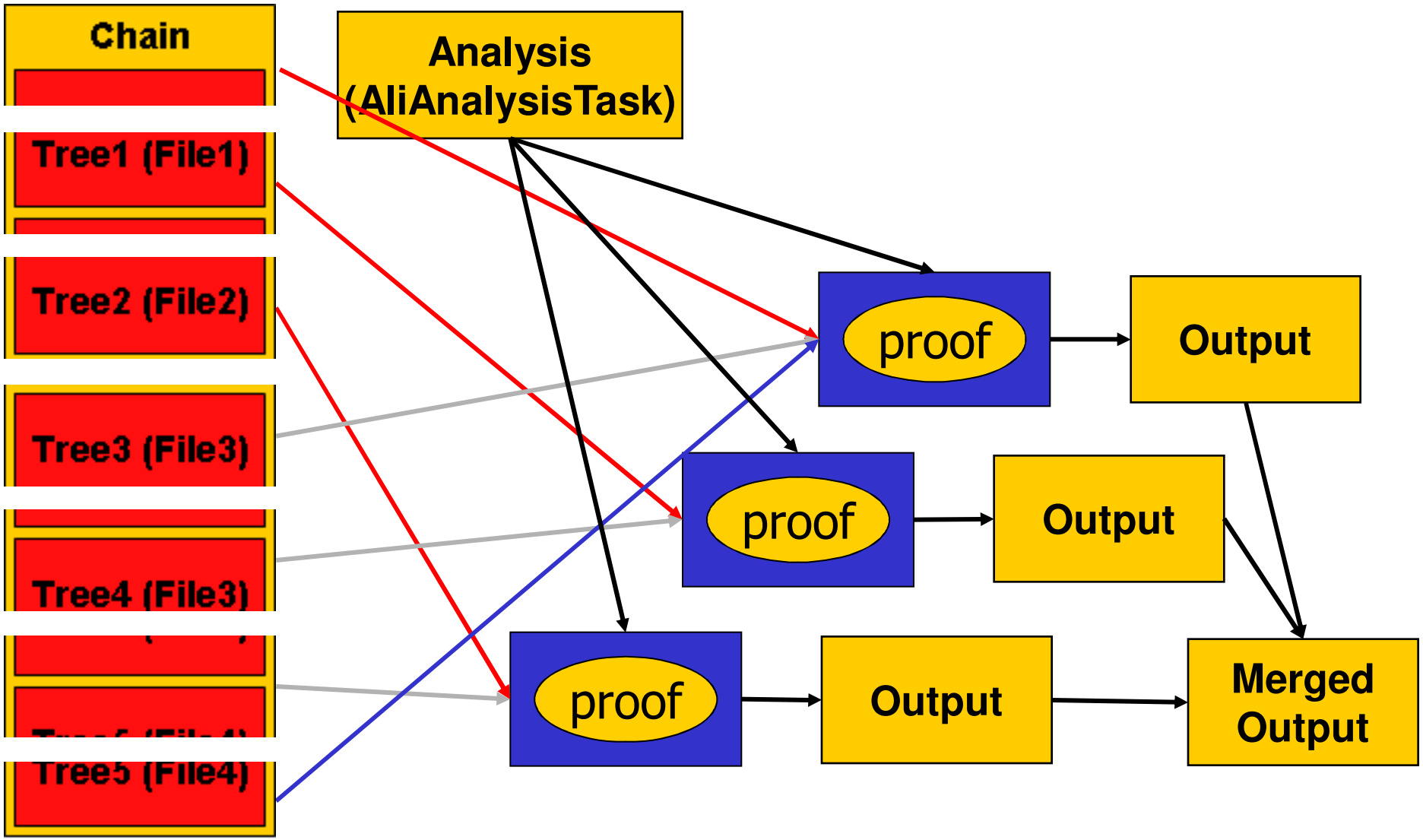
- The analysis runs on several slaves, therefore partial results have to be merged
- Objects are identified by name
- Standard merging implementation for histograms available
- Other classes need to implement **Merge(TCollection*)**
- When no merging function is available all the individual objects are returned



Workflow Summary



Workflow Summary



Packages

- PAR files: **P**ROOF **AR**chive. Like Java jar
  - Gzipped tar file
  - PROOF-INF directory
    - BUILD.sh, building the package, executed per slave
    - SETUP.C, set environment, load libraries, executed per slave
- API to manage and activate packages
  - UploadPackage("package")
  - EnablePackage("package")

CERN Analysis Facility

- The **C**ERN **A**nalysis **F**acility (CAF) will run PROOF for ALICE
  - Prompt analysis of pp data
  - Pilot analysis of PbPb data
  - Calibration & Alignment
- Available to the whole collaboration but the number of users will be limited for efficiency reasons
- Design goals
  - 500 CPUs
  - 100 TB of selected data locally available

Evaluation of PROOF

- CAF1 since May 2006
  - 40 machines, 2 CPUs each, 200 GB disk each
- CAF2 since Oct 2008
  - 14 machines, 8 cores each, 2.33 TB disk each
- CAF3 soon
  - 45 machines, 8 cores, 3 TB disk each

Hands-On

- Getting ready...
- Run a task that accesses ESD
  - Locally
  - PROOF
  - Modify it...
- Run a task that accesses MC
  - PROOF
- Reading log files, resetting session, etc.
- What about “interactive” grid ?
  - AliEn plug-in hands on

Warm up

- Log into LXPLUS with your account
- Preconditions
  - Use bash shell (type "bash")
  - Grid certificate (usercert.pem/userkey.pem) in ~/.globus
    - Howto: convert from .p12 to .pem
      - `openssl pkcs12 -clcerts -nokeys -out usercert.pem -in cert.p12`
      - `openssl pkcs12 -nocerts -out userkey.pem -in cert.p12`
  - On the tutorial page http://aliceinfo.cern.ch/Offline/AliRoot/Manual.html,
    - Save "*Files for the PROOF tutorial (tgz)*" to your home dir and extract it
- Set up environment
  - Execute the command
    `source /afs/cern.ch/alice/caf/caf-lxplus.sh -alien v4-19-04-AN`
  - You will be prompted for your certificate password
- Check ROOT
  - Start it. Does it show ROOT version 5.26/00?

Files to be used

- AF-v4-19-04-AN.par
  Par archive for PDC10 data and analysis framework
- AliAnalysisTaskPt.{cxx,h}
  Task that creates an uncorrected pT spectrum from ESD tracks
- AliAnalysisTaskPtMC.{cxx,h}
  Task that creates an pT spectrum from the MC particles
- files.txt
  Short list of MC files to test the analysis task locally

Run a task locally

- Start ROOT
- Try the following lines and once they work add them to a macro run.C (enclose in {})
- Load needed libraries
  - gSystem->Load("libTree.so");
  - gSystem->Load("libGeom.so");
  - gSystem->Load("libVMC.so");
  - gSystem->Load("libSTEERBase.so");
  - gSystem->Load("libESD.so");
  - gSystem->Load("libAOD.so");
  - gSystem->Load("libANALYSIS.so");
  - gSystem->Load("libANALYSISalice.so");
- Add the AliRoot include path (only needed for local case)
  - gROOT->ProcessLine(".include $ALICE_ROOT/include");

Run a task locally (2)

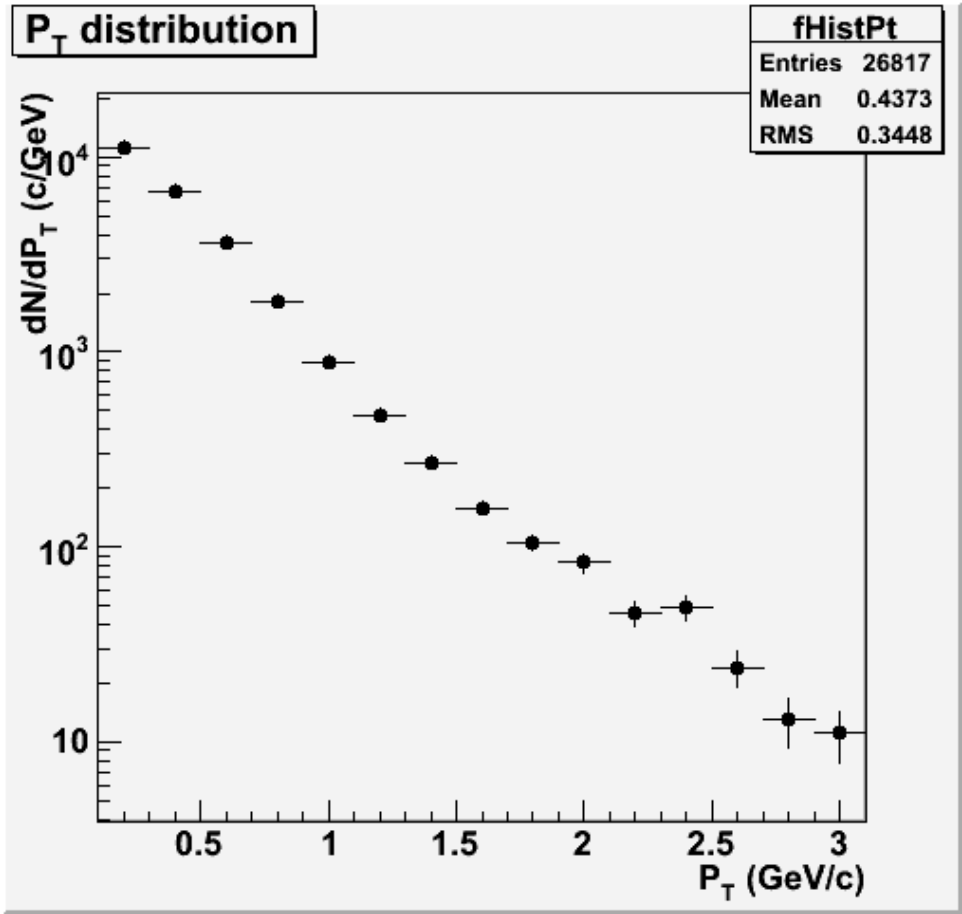
- Create the analysis manager
  - mgr = new AliAnalysisManager("testAnalysis");
- Create the analysis task and add it to the manager
  - gROOT->LoadMacro("AliAnalysisTaskPt.cxx++g");
    - "+" means compile; "g" means debug
  - task = new AliAnalysisTaskPt("TaskPt");
  - mgr->AddTask(task);
- Add the ESD handler (to access the ESD)
  - esdH = new AliESDInputHandler;
  - mgr->SetInputEventHandler(esdH);
- Add the lines to the macro run.C

Run a task locally (3)

- Create a chain
  - gROOT->LoadMacro("$ALICE_ROOT/PWG0/CreateESDChain.C");
  - chain = CreateESDChain("files.txt");
- Attach the input (the chain)
  - cInput = mgr->GetCommonInputContainer();
  - mgr->ConnectInput(task, 0, cInput);
- Create a place for the output (a histogram: TH1)
  - cOutput = mgr->CreateContainer("cOutput", TList::Class(), AliAnalysisManager::kOutputContainer, "Pt1.root");
  - mgr->ConnectOutput(task, 1, cOutput);
- Enable debug (optional)
  - mgr->SetDebugLevel(2);
- Add the lines to the macro run.C

Run a task locally (4)

- Initialize the manager
  - mgr->InitAnalysis();
- Print the status (optional)
  - mgr->PrintStatus();
- Run the analysis
  - mgr->StartAnalysis("local" , chain);
- Add the lines to the macro run.C
- After running look at the output and check the content of the file Pt1.root



PROOF - Connecting and Packages

- Connecting to the PROOF cluster
  - gEnv->SetValue("XSec.GSI.DelegProxy", "2");
  - TProof::Open(”alicecaf");
- Managing packages
  - Upload (= copy to the cluster)
    - gProof->UploadPackage(“AF-v4-19-04-AN");
  - Enable (= compile)
    - gProof->EnablePackage("AF-v4-19-04-AN");
  - Clean (= remove)
    - gProof->ClearPackage("AF-v4-19-04-AN");
    - Known issue on AFS: Removal may fail. Try again after few seconds…
  - Clean all (in case some libraries are messed up)
    - gProof->ClearPackages();

PROOF datasets

- A dataset represents a list of files (e.g. physics run X)
  - Correspondence between AliEn collection and PROOF dataset
- Users register datasets
  - The files contained in a dataset are automatically staged from AliEn (and kept available)
  - Datasets are used for processing with PROOF
    - Contain all relevant information to start processing (location of files, abstract description of content of files)
- Datasets are public for reading, common datasets are available (for data of common interest)
- Learn about dataset at
  - http://aliceinfo/Offline/Activities/Analysis/CAF

Datasets in Practice

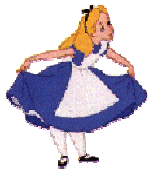
- Upload to PROOF cluster
  - gProof->RegisterDataSet("myDataSet", proofColl);
- Check status
  - gProof->ShowDataSets();

```
root [3] gProof->ShowDataSets()
Dataset URI                                |# Files|Default tree|# Events|   Disk  |Staged
/default/jgrosseo/ESD100                   |    6764|/esdTree    |   676400|  343 GB|  100 %
/default/jgrosseo/run82XX_part1            |   10000|/esdTree    |   998900|  288 GB|   99 %
/default/jgrosseo/run82XX_part2            |   10000|/esdTree    |   944700|  272 GB|   94 %
/default/jgrosseo/run82XX_part3            |   10000|/esdTree    |   987900|  285 GB|   98 %
/default/jgrosseo/ESD600                   |    1844|/esdTree    |   184400|   51 GB|  100 %
/default/jgrosseo/ESD_FullMisalignment     |     944|/esdTree    |    92100|   47 GB|   97 %
/default/jgrosseo/run12000                 |      62|/esdTree    |       49|    4 GB|   79 %
/default/jgrosseo/ESD5000                  |   21899|/esdTree    |  2189800| 1096 GB|   99 %
/default/jgrosseo/ProofSessionFiles        |   25438|/esdTree    |  4460900| 1640 GB|  100 %
```

- Use it
  - gProof->Process("myDataSet", "TMySelector.cxx+");

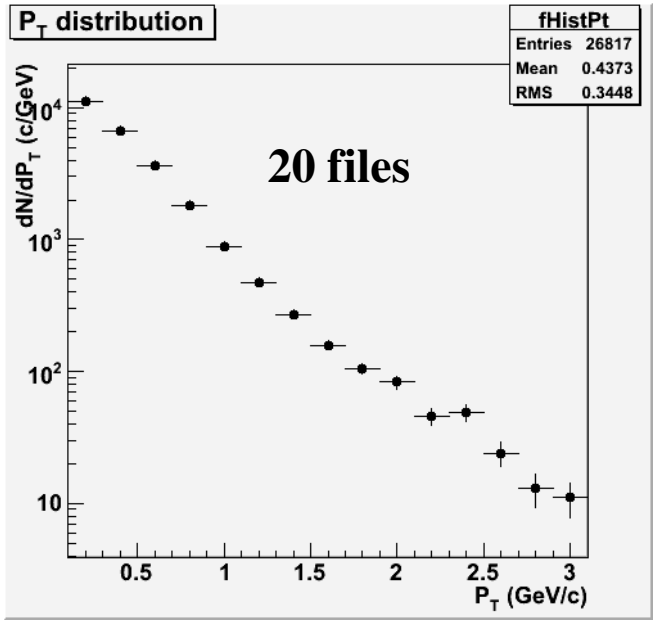
Running a task in PROOF

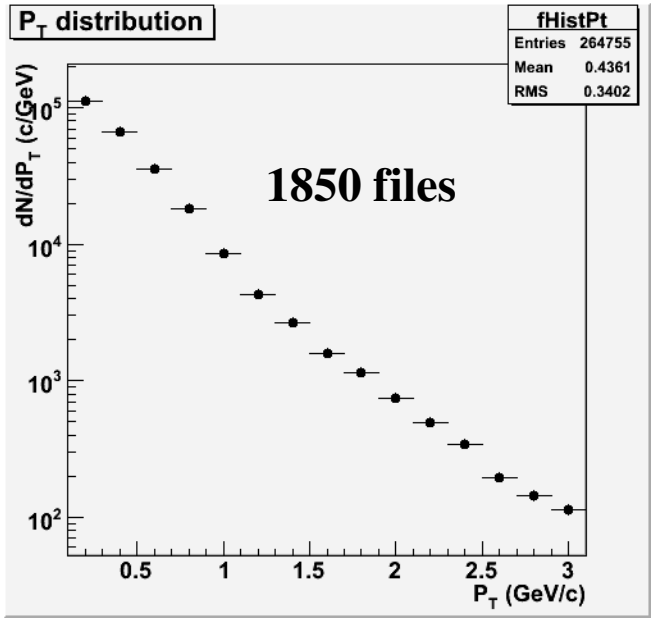
- Copy run.C to runProof.C
- Add connecting to the cluster
  - gEnv->SetValue("XSec.GSI.DelegProxy", "2");
  - TProof::Open(”alicecaf");
- Replace the loading of the libraries with uploading the packages
  - gProof->UploadPackage("AF-v4-19-04-AN");
  - gProof->EnablePackage("AF-v4-19-04-AN");
- Replace the loading of the task with
  - gProof->Load("AliAnalysisTaskPt.cxx++g");



Running a task in PROOF (2)

- Replace in StartAnalysis
  - "local" with "proof"
  - The chain with dataset "/COMMON/COMMON/LHC09d10_run10482X"
- Change output file name to Pt2.root
- Add only 10000 entries to be processed
  - As last parameter of StartAnalysis()
- Run it!





Progress dialog

New ROOT has a fancy speedometer

PROOF Query Progress: jgrosseo@lxb6046.cern.ch

Executing on PROOF cluster "lxb6046.cern.ch" with 33 parallel workers:

Selector: TMySelector.cxx

100 files, number of events 10000, starting event 0

Initialization time: 0.9 secs

Processed: 10000 events (1790.14 MBs) in 4.9 sec

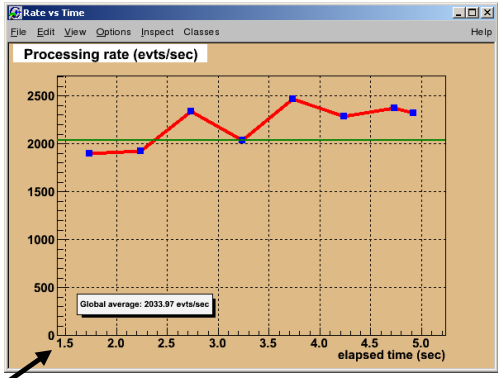
Processing rate: 2034.0 evts/sec (364.1 MBs/sec)

Query statistics

Close dialog when processing is complete

Show only logs from query last

Stop | Cancel | Close | Show Logs | Rate plot



Abort query and view results up to now

Abort query and discard results

Show log files

Show processing rate

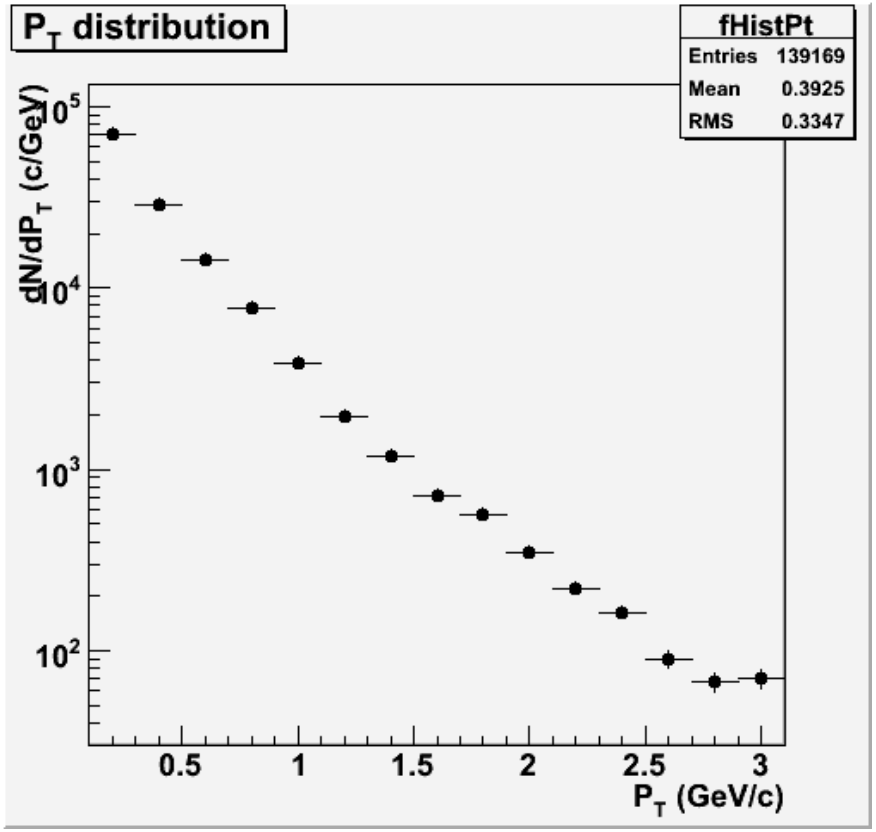
Looking at the task

- Constructor
  - Called once when the task is created
  - Input/Output is connected
- UserCreateOutputObjects
  - Called once per slave
  - Create histograms
- UserExec
  - Called once per event
  - Track loop, tracks are counted, histogram filled, output "posted"
- Terminate
  - Called once on the client (your laptop/PC)
  - Histogram read back from the output stream, visualized, saved to disk

Changing the task

- Add a $|\eta| < 0.5$ cut

```
Float_t eta = track->Eta();
if (TMath::Abs(eta) > 0.5)
    continue;
```

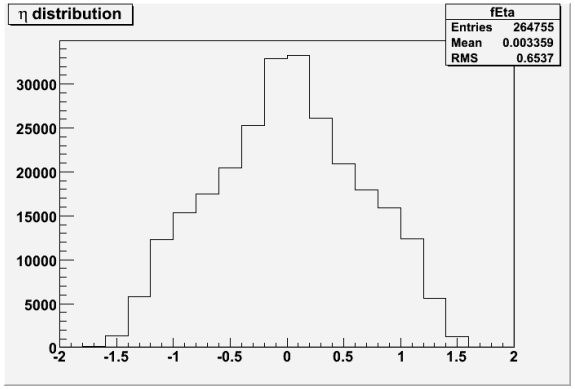


Changing the task (2)

- Add a second plot: η distribution
  - Header file (.h file)
    - Add new member: TH1F* fEta; // eta distribution
  - Constructor
    - Initialize member: fEta(0)
  - UserCreateOutputObjects
    - Create histogram
      fEta = new TH1F("fEta", "#eta distribution", 20, -2, 2);
    - Add the histogram to the output list
      fOutputList->Add(fEta);
  - UserExec
    - Get η like in previous example
    - Fill histogram: fEta->Fill(eta);

Changing the task (3)

- Terminate
  - Read histogram from the output slot
    fEta = dynamic_cast<TH1F*>
    (fOutputList->FindObject("fEta"))
  - Introduce an if statement to check if the object was retrieved
    if (!fEta) {
    Printf("ERROR: fEta was not found");
    return;
    }
  - Draw the histogram
    new TCanvas;
    fEta->DrawCopy();

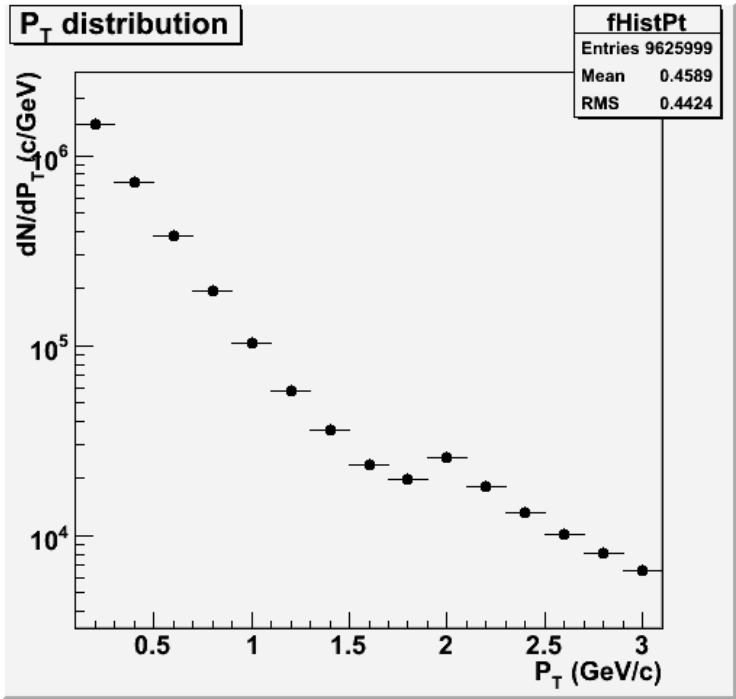


Read Monte Carlo tracks

- Use task AliAnalysisTaskPtMC.{h,cxx}
- Copy runProof.C to runProofMC.C
- Change AliAnalysisTaskPt to AliAnalysisTaskPtMC
- Add access to the MC event handler

```
handler =
   new AliMCEventHandler;
mgr->SetMCtruthEventHandler
   (handler);
```

- Change output filename to PtMC.root
- Run it!



Looking at the MC task

- Very similar to ESD track case
- Instead of looping over content of fESD, MC event is retrieved by
  - AliMCEvent* mcEvent = MCEvent();
    if (!mcEvent) {
      Printf("ERROR: Could not retrieve MC event");
      return;
    }

Reading log files

- When your task crashes
- You can access the output of the last query by clicking on the “Show Log” button in the PROOF progress window
- You can retrieve the output from any previous query
  - Open ROOT
  - Get a PROOF manager object
    mgr = TProof::Mgr(”alicecaf")
  - Get the log files from the last session
    logs = mgr->GetSessionLogs(0) // 0=last query
  - Display them
    logs->Display()
  - Search for a special word (e.g. segmentation violation)
    logs->Grep("segmentation violation")
  - Save them to a file
    logs->Save("*", "logs.txt")

Some Goodies…

- Resetting environment
  - TProof::Reset(”alicecaf")
  - TProof::Reset(”alicecaf", kTRUE)
- Compile with debug
  - Load("<task>+g")
- Create a package from AliROOT
  - make ESD.par

References

- More information on http://aliceinfo.cern.ch/Offline/Activities/Analysis/CAF
- Read the FAQ on the webpage above
- Please join the mailing list **alice-project-analysis-task-force@cern.ch** by going to http://listboxservices.web.cern.ch/listboxservices