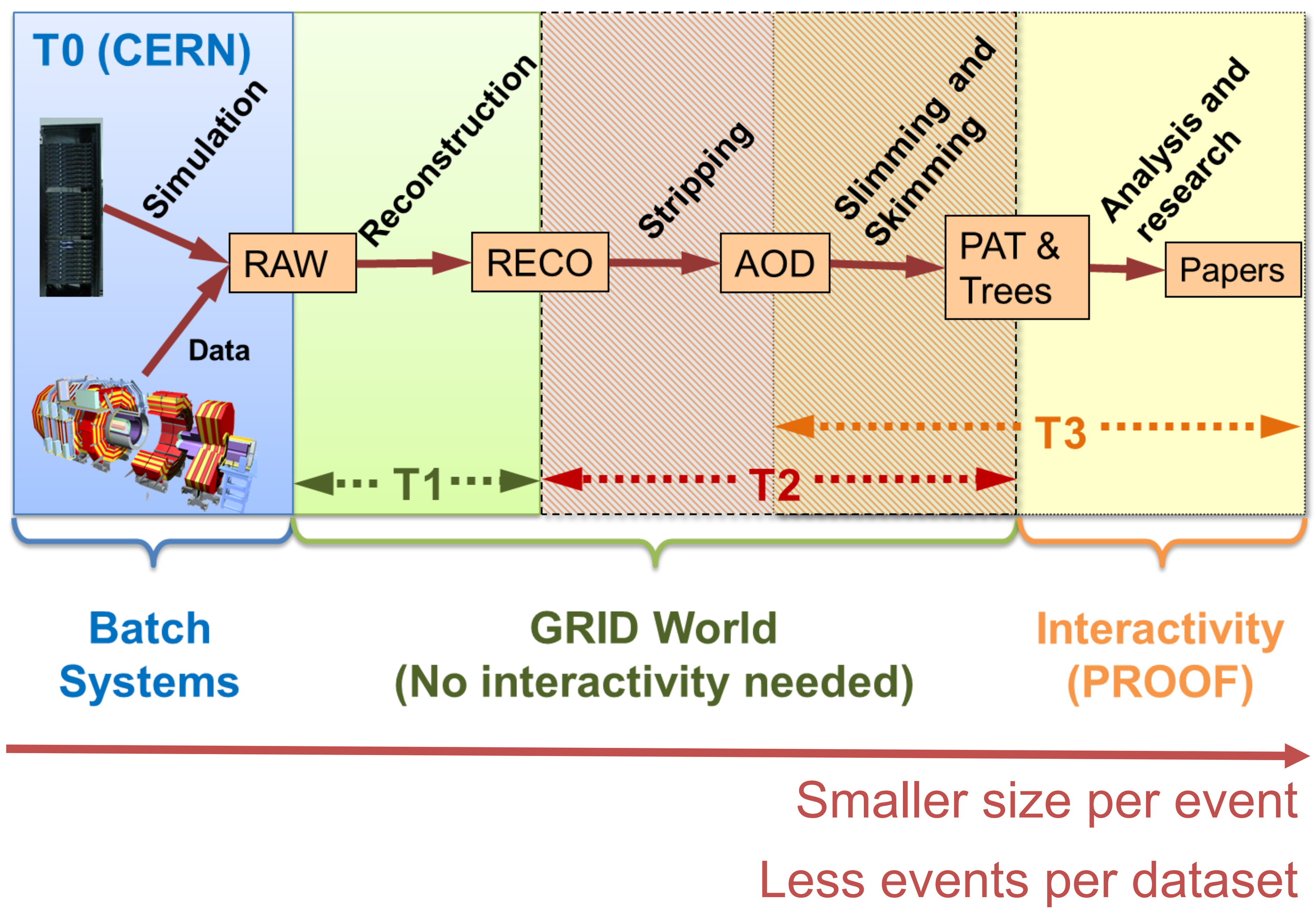
A PROOF Analysis Framework

I. González Caballero[1,3], A.Y. Rodríguez Marrero[2],
E. Fernández del Castillo[2], A. Cuesta Noriega[1]

[1]Universidad de Oviedo - Spain, [2]Instituto de Física de Cantabria (UC-CSIC) - Spain

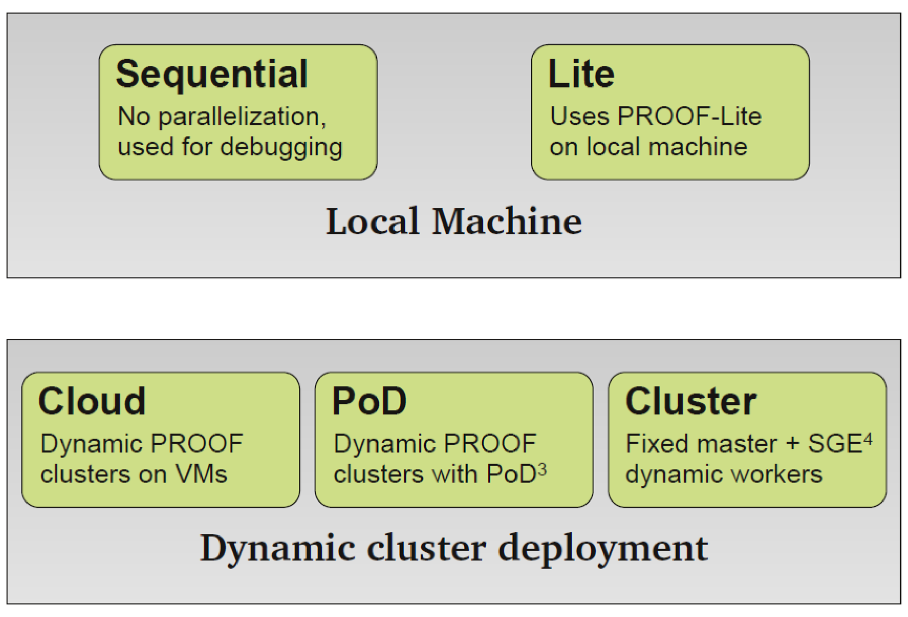
[3]*Isidro.Gonzalez.Caballero@cern.ch*

Evolution of data formats in a typical LHC experiment



As data evolves higher level physics objects are added to the structure. Other operations reduce the amount of data that a given analysis needs to process in the end.

PROOF, computer farms and batch systems



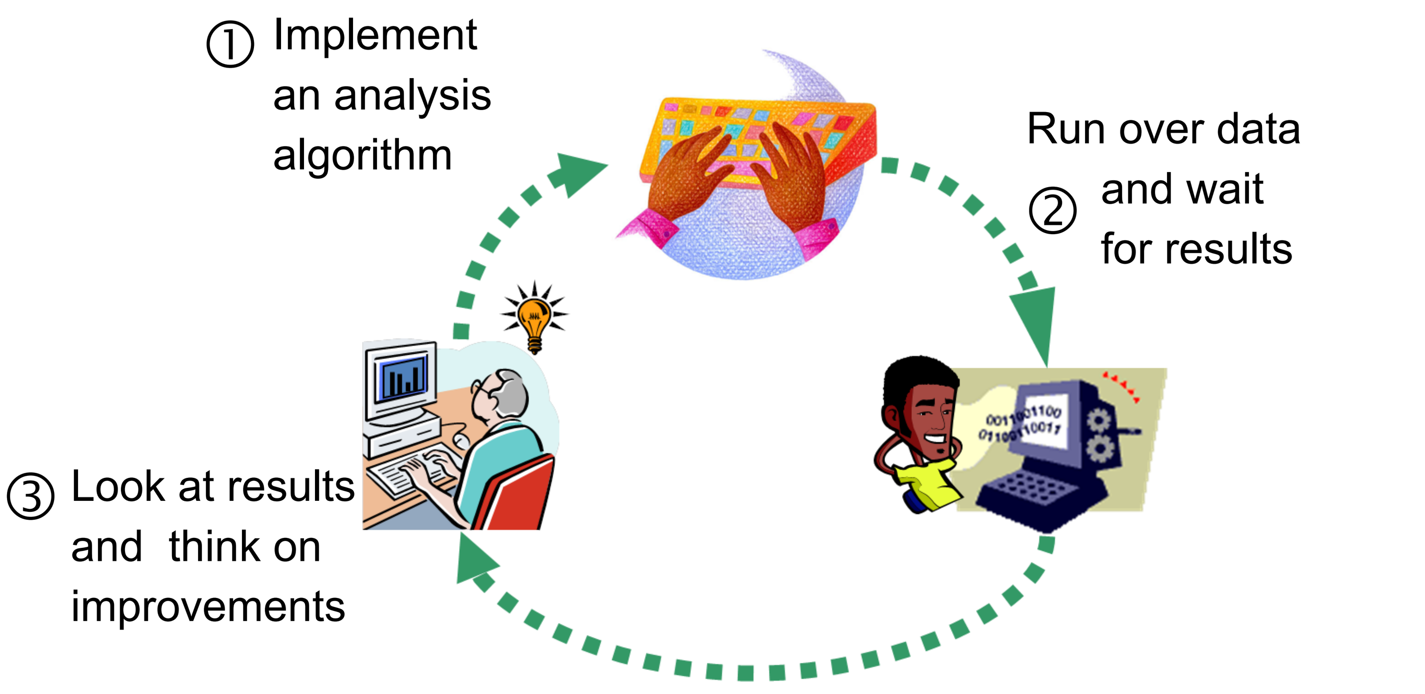
The Parallel ROOT Facility provides an efficient and transparent way to process ROOT trees distributedly. Clusters of computers like Tier-2 and Tier-3 centres can be used to improve processing performance. Batch and Cloud systems can be used to build PROOF clusters dynamically by using some specific tools: PROOF Cluster, PoD, Cloud Cluster,...

Modern many-core desktops or laptops can also benefit from PROOF using the PROOF-Lite mode.

Being able to run in sequential mode is important for debugging.

For more details see poster: *Integrating PROOF Analysis in Cloud and Batch Clusters*

The data analysis development cycle



Some sort of interactivity is desirable so the developer time is spent thinking rather than waiting for results. To achieve this a large amount of CPU is needed during short periods of time.

PROOF Analysis Framework design goals

1. Physicists should concentrate on building the analysis rather than on the computing technicalities
2. Migrating traditional ROOT based analysis code should be easy
3. Code should be valid across the different PROOF modes supported
4. The need for special configurations on the computing infrastructures used should be kept minimal

PAF achieves these objectives through an automatically generated `TSelector` subclass that contains all the tree information. Another class managing all the interaction with PROOF exposes only the configuration settings.

PAF — Analysis code

The analysis code is implemented in a subclass ❶ of the automatically generated PAF selector. Four virtual methods can be specialised to:

- Read input parameters
- Initialise analysis result objects
- Do the appropriate calculation at each event and update analysis results
- Perform final operations on results

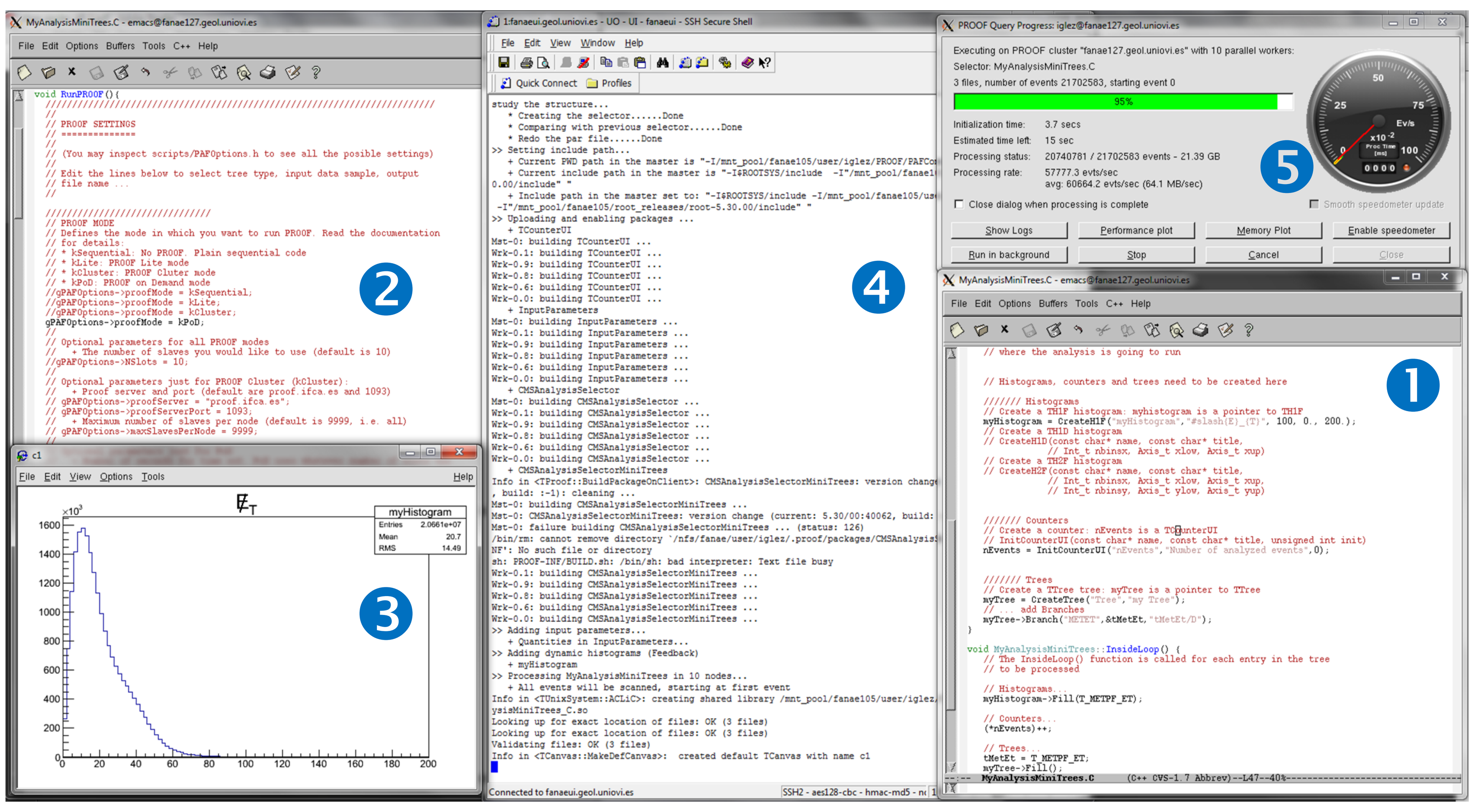
Analysis result data is automatically saved to a ROOT file for later inspection.

PAF — PROOF Configuration

A macro file ❷ starts the PROOF session by setting:

- PROOF Mode, number of workers to use,...
- Input data files
- Output file where results are to be stored
- Dynamic histograms ❸: Histograms that will be updated as they get filled
- Additional analysis packages hosting modularised code (ex. official electron selection)
- Input parameters for the analysis
- Selector containing the analysis code

PAF Example session



A typical PAF session runs from inside ROOT invoking the configuration macro from a normal terminal ❹.

PAF starts by setting (or reusing if possible) a PROOF session according to the chosen mode.

It then transparently generates, compiles, packages and uploads to the workers all the code that needs it.

At this point a window ❺, showing the progress of the event processing and providing access to peformance information and logs, is launched.

Universidad de Oviedo
Instituto de Física de Cantabria
Universidad de Cantabria
CSIC Consejo Superior de Investigaciones Científicas