CAF / Proof

ALICE Users Experiences

Hélène Ricaud

Institut Pluridisciplinaire Hubert Curien
- Strasbourg -

CAF / Proof Workshop
December 2007 - Geneva, CERN -

IPHC
Institut Pluridisciplinaire Hubert CURIEN STRASBOURG

Why we have chosen to use CAF / Proof

➢ **It allows very fast development cycles:**

- Given the complexity of the GRID system, CAF was an easier way to develop and debug our analysis codes.

- It is possible to run an analysis and see quickly the results.

➢ **Portability of the analysis codes:**

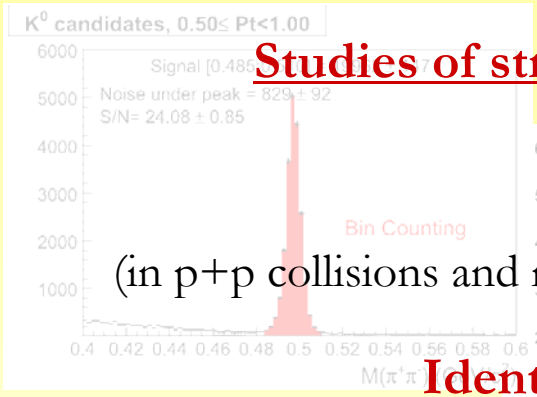
The codes developed on CAF with Proof can run on the GRID as well without any modifications.

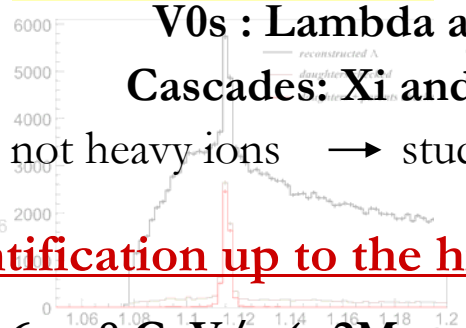
➢ **It has even been decided to install Proof locally in our laboratory (in progress)**

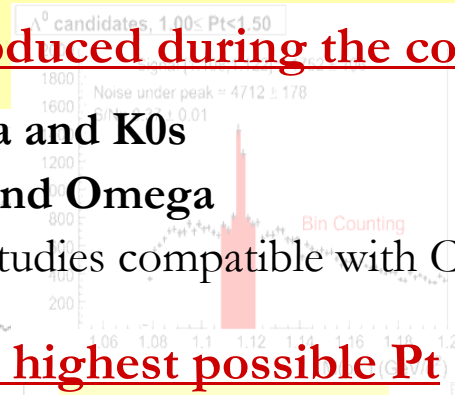
Since the number of machines we have is going to increase soon, we have thought about installing Proof.

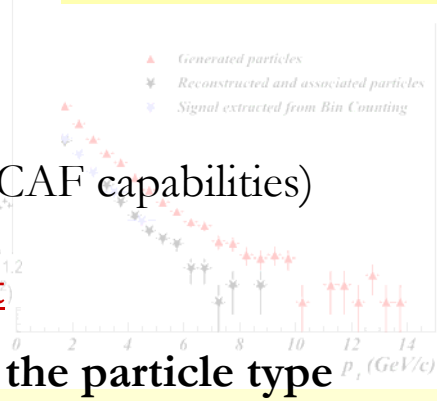
In the future, we will use both: CAF and processing with Proof locally.

What we want to do on CAF: our analysis









Studies of strange particles produced during the collision:

V0s : Lambda and K0s

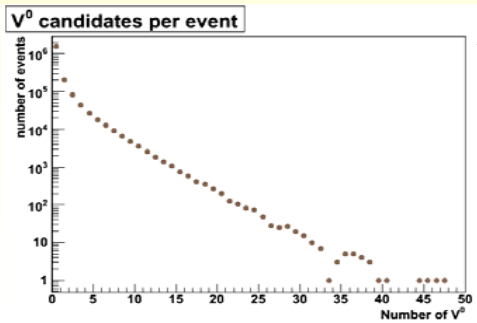
Cascades: Xi and Omega

(in p+p collisions and not heavy ions → studies compatible with CAF capabilities)

Identification up to the highest possible Pt

which means around 6 or 8 GeV/c (~2M events) depending on the particle type

We need to loop over quite a lot of input data files (AliESD files)

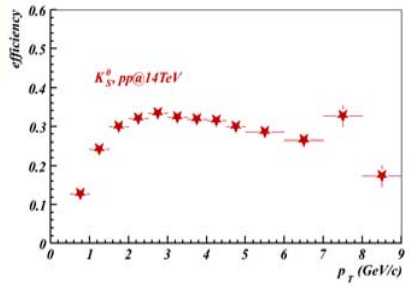


The number of candidates per event and, above all, the signal is not very high (especially for cascade particles).

Getting enough statistic to study particles in each Pt bin requires to loop over many AliESD files...

We need access to Monte Carlo information

In order to evaluate the reconstruction efficiency, we ask for Monte Carlo information (gAlice and Kinematics files) which increases significantly the processing time.



Our analysis, step by step

➢ *Quite easy setup !*

Quick startup thanks to a nice dedicated web page

(http://aliceinfo.cern.ch/Offline/Analysis/CAF)

The script to get the same Root versions on CAF and lxplus nodes (mandatory since the code is compiled twice, on CAF and locally) is even provided !!!

(**afs/cern.ch/alice/caf/caf-lxplus.sh**)

Although we can stage files from Alien, we still haven't use that option ourselves. Up to now, we have always looped on events already distributed on CAF.

➢ *Reading PDC06 events: AliSelector (1 year ago !)*

We have started by analysing the PDC06 events available (around 2 M events):

- macro written as an **AliSelector** which derived from the **TSelector**
- accessing the ESD files (Reconstruction output files) required to upload an ESD package by standard Proof functionality.

Our analysis, step by step

➢ *Accessing Monte Carlo information with an AliSelector*

But the analyses we wanted to do require access to Monte Carlo information. In ALICE environment this is done through the so called **RunLoader**.

- macro written as an **AliSelectorRL**;
- due to complex dependences, it requires the full ALICE software framework.

Need to enable full AliRoot on CAF on every worker and to get the same AliRoot version on lxplus nodes !

➢ *Integration to ALICE's new Analysis Framework (PDC07 events)*

Framework that combines several analyses in order to loop only one time on each ESD files.

→ "Analysis train"

- macro written as an **AliAnalysisTask** (based on TSelector)
- changes in the way we access Kinematics files: dependences have been reduced.

→ possibility to create a package (STEERbase) that is loaded and enabled on each worker.
No need of full AliRoot !

Analysis Framework concept = Will for making the use of Proof transparent for users

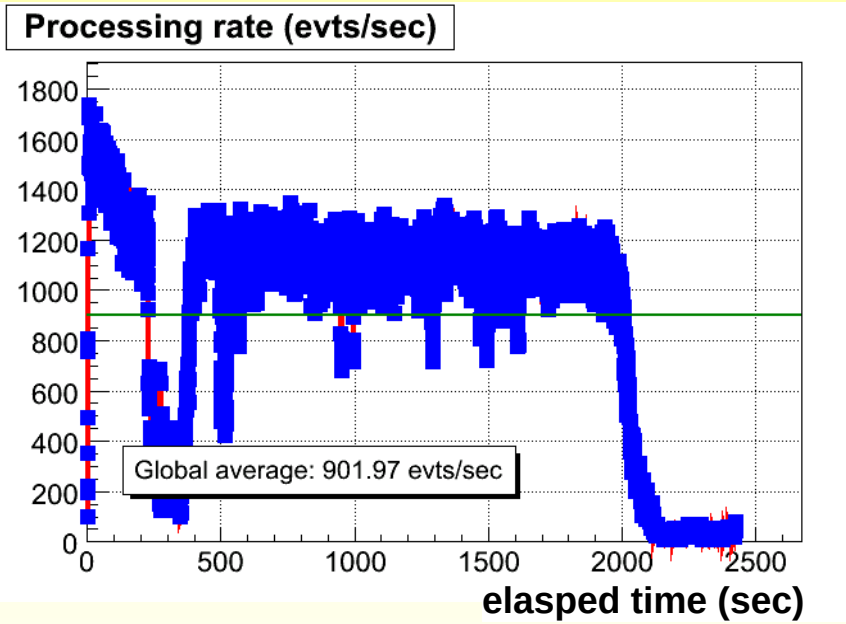
User's code is not dependent on the system on which it runs

⊆ just a switch to do !!

```
AliAnalysisManager *mgr = new AliAnalysisManager ();
mgr->StartAnalysis("proof",chain);  or  mgr->StartAnalysis("local",chain);
```

Our analysis, step by step

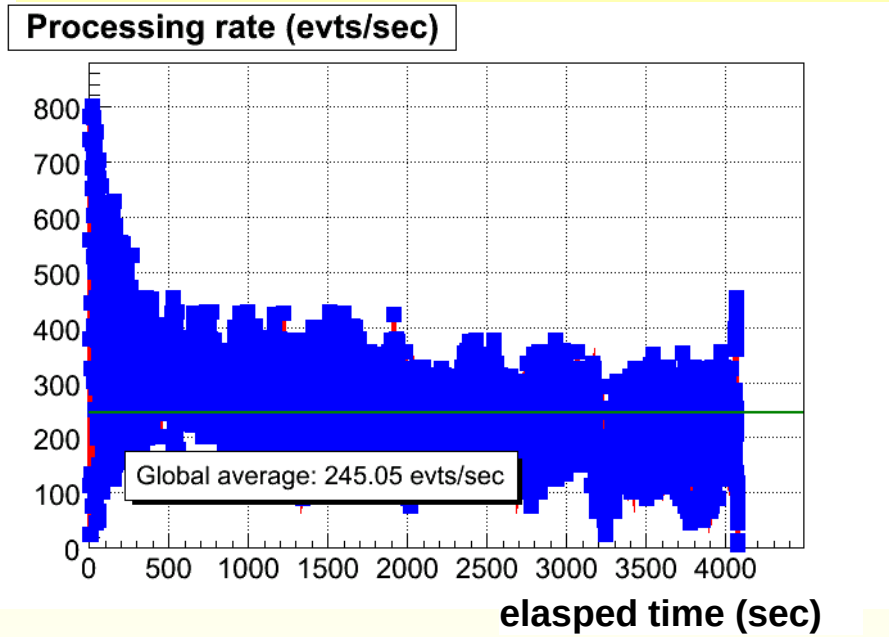
Our macro written as an AliSelectorRL



Mst-0: grand total: sent 120 objects, size: 334050 bytes
Real time 0:44:59, CP time 109.190

Input: ~ 2 M of PDC06 events
Output: ~ 90 TH1F, ~ 30 TH2F

Our macro written as an AliAnalysisTask



Mst-0: grand total: sent 2 objects, size: 103681 bytes
Real time 1:10:35, CP time 5.930

Input: 1 M of PDC07 events
Output: ~ 30 TH1F, ~ 20 TH2F

Processing time much higher in the case of AliAnalysisTask…
But what costs the most is probably the time to access the data (PDC06 files are on CAF machines whereas we access PDC07 files via staging from Alien)

Disadvantages & Difficulties encountered

- Choosing, building, re-building the needed packages since the aim is not to deploy full AliRoot on CAF.
→ advantage: modularity but it requires time to upload the packages…

- Synchronization between Root/AliRoot version on lxplus and CAF.
→ constrains for users.

- CAF efficiency seems to decrease rapidly with the number of users.

- As soon as one of the workers is removed from the active list, the processing rate starts decreasing and as a result the job will never come to an end.

- A problem that appears during the execution of one user's job (memory exceeded, memory leaks…) can apparently affect others user jobs (especially if one worker is removed)…

- Running over a long chain fails with the new Analysis Framework (maximum of events we've managed to analyse: 1 M).
Memory leak inside the Task Manager ? Problem related to Proof ?

Users requests

What could be implemented:

- An easy way to access log files.
- A system scheduler based on user history, resources needed.
- A memory monitoring to profile the memory consumption, since it seems to be one of the main problems encountered by users.