Experience with ROOT for ALICE analyses

Jochen Klein
for the ALICE Collaboration

CERN

"ROOT Turns 20" Users' Workshop

Saas-Fee, Switzerland

September 15th, 2015

Outline

- usage of ROOT in ALICE
- organization of analyses in ALICE
- some "annoyances" of ROOT
- potential "ameliorations" from our perspective

not a laudatio but:
constructive criticism
as input for consideration

Disclaimer

not a report from the ALICE offline or online group,
but: **experience from the user/analysis perspective**

most users are physicists **without deep computing background**

thanks to everybody who provided comments/suggestions

ROOT in ALICE

- raw data from the experiment packed into ROOT objects
- experimental parameters and calibration results stored in Offline Condition DataBase (OCDB) as `TFiles`
- reconstruction transforms raw data into Event Summary Data (ESD) `TTrees`
- filtering data structures for analysis into Analysis Oriented Data (AOD) `TTrees`
- user analysis on ESD/AOD `TTree`

  skimmed data (nanoAODs) help for some analyses, but many analyses would not benefit

---

we are still with ROOT 5

Analyses in ROOT

typical workflow (with iterations)

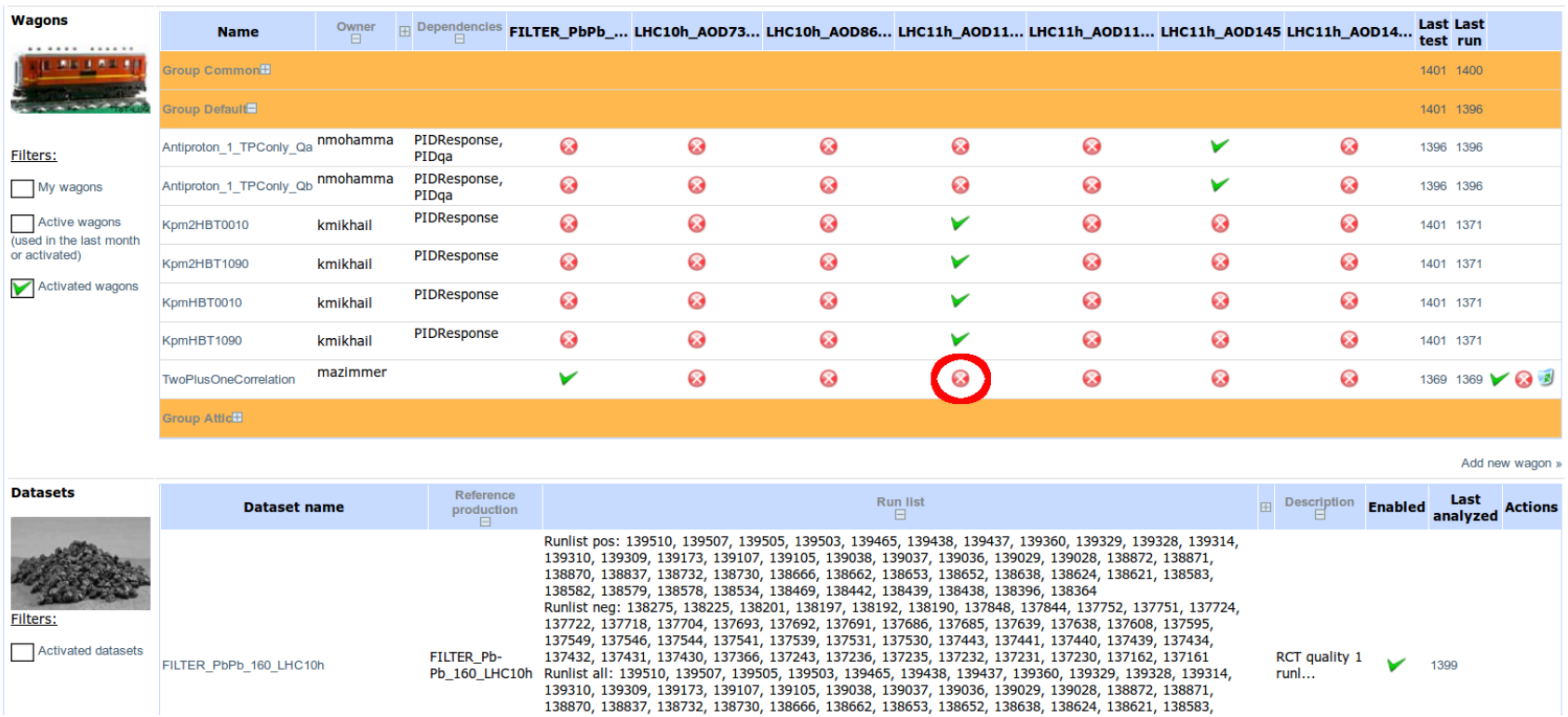
- analysis (`TTask`) run on chain of ESD or AOD trees
  typical output: histograms, sometimes trees
  (possibly more trees in the future)
- (local) post-processing
- produce physics plots

technical realization

- custom-built framework around existing components
- users commit code to repository,
  deployed to cvmfs and grid sites
- analysis task typically run as part of a train,
  i.e. running collection of tasks from different users on the same data

LEGO train

- trains mostly run on the Grid, split into subjobs
- merge output from individual jobs
- highly automated system, steered through web interface, includes custom-built monitoring, profiling, and book-keeping



Wagons

Filters:
- [ ] My wagons
- [ ] Active wagons (used in the last month or activated)
- [x] Activated wagons

Name	Owner	Dependencies	FILTER_PbPb_...	LHC10h_AOD73...	LHC10h_AOD86...	LHC11h_AOD11...	LHC11h_AOD11...	LHC11h_AOD145	LHC11h_AOD14...	Last test	Last run
Group Common										1401	1400
Group Default										1401	1396
Antiproton_1_TPConly_Qa	nmohamma	PIDResponse, PIDqa	✗	✗	✗	✗	✗	✓	✗	1396	1396
Antiproton_1_TPConly_Qb	nmohamma	PIDResponse, PIDqa	✗	✗	✗	✗	✗	✓	✗	1396	1396
Kpm2HBT0010	kmikhail	PIDResponse	✗	✗	✗	✓	✗	✗	✗	1401	1371
Kpm2HBT1090	kmikhail	PIDResponse	✗	✗	✗	✓	✗	✗	✗	1401	1371
KpmHBT0010	kmikhail	PIDResponse	✗	✗	✗	✓	✗	✗	✗	1401	1371
KpmHBT1090	kmikhail	PIDResponse	✗	✗	✗	✓	✗	✗	✗	1401	1371
TwoPlusOneCorrelation	mazimmer		✓	✗	✗	✗	✗	✗	✗	1369	1369
Group Attic											

Add new wagon »

Datasets

Filters:
- [ ] Activated datasets

Dataset name	Reference production	Run list	Description	Enabled	Last analyzed	Actions
FILTER_PbPb_160_LHC10h	FILTER_Pb-Pb_160_LHC10h	Runlist pos: 139510, 139507, 139505, 139503, 139465, 139438, 139437, 139360, 139329, 139328, 139314, 139310, 139309, 139173, 139107, 139105, 139038, 139037, 139036, 139029, 139028, 138872, 138871, 138870, 138837, 138732, 138730, 138666, 138662, 138653, 138652, 138638, 138624, 138621, 138583, 138582, 138579, 138578, 138534, 138469, 138442, 138439, 138438, 138396, 138364 Runlist neg: 138275, 138225, 138201, 138197, 138192, 138190, 137848, 137844, 137752, 137751, 137724, 137722, 137718, 137704, 137693, 137692, 137691, 137686, 137685, 137639, 137638, 137608, 137595, 137549, 137546, 137544, 137541, 137539, 137531, 137530, 137443, 137441, 137440, 137439, 137434, 137432, 137431, 137430, 137366, 137243, 137236, 137235, 137232, 137231, 137230, 137162, 137161 Runlist all: 139510, 139507, 139505, 139503, 139465, 139438, 139437, 139360, 139329, 139328, 139314, 139310, 139309, 139173, 139107, 139105, 139038, 139037, 139036, 139029, 139028, 138872, 138871, 138870, 138837, 138732, 138730, 138666, 138662, 138653, 138652, 138638, 138624, 138621, 138583,	RCT quality 1 runl...	✓	1399	

typical train run: 10 k jobs, $\sim$ 1 year total CPU time

Run 2, Run 3, ...

- ROOT will stay omnipresent in the data chain
- for Run 3:
  merge online and offline $\rightsquigarrow$ O^2
- full online calibration
- will involve heavy streaming of data between processes
  ROOT serialization appears to be slower than boost, protobuf, ...
- possibility of simplified streaming without schema evolution?
  for optional use in online components

So much for introduction, now to feedback

Class hierarchy

- class hierarchy not very intuitive and modern
  for people knowing C++
  (e.g. histogram hierarchy, namespacing, templating)
- new people get used to ROOT class hierarchies
  instead of modern design
- template usage could simplify many things,
  also an ALICE problem: was discouraged for a long time
- by now significant overlap with STL,
  but no clear recommendations when to use what

---

hard to change in a backward compatible way (break it?),
but explanations or recommendations would help

Documentation

- quality of documentation varies widely
- some parts are excellent,
  for other parts it is hard to understand the intended usage
- often hard to find **reliable** information
  on what is being done in mathematical terms
  (e.g. uncertainties for efficiency calculation),
  or range of applicability

Example

excellent RooFit manuals, howtos, . . . ,

but minimization factory in ROOT hard to understand
minuit, minuit2, . . .

Communication of new developments

- takes time for new developments to arrive in community
- often not obvious what's the intended use is
- in-code documentation often does not explain concepts or intended use

---

- communication could be improved, e.g.

  through update notes (à la application note) or experiment representatives
- add "Best practices"

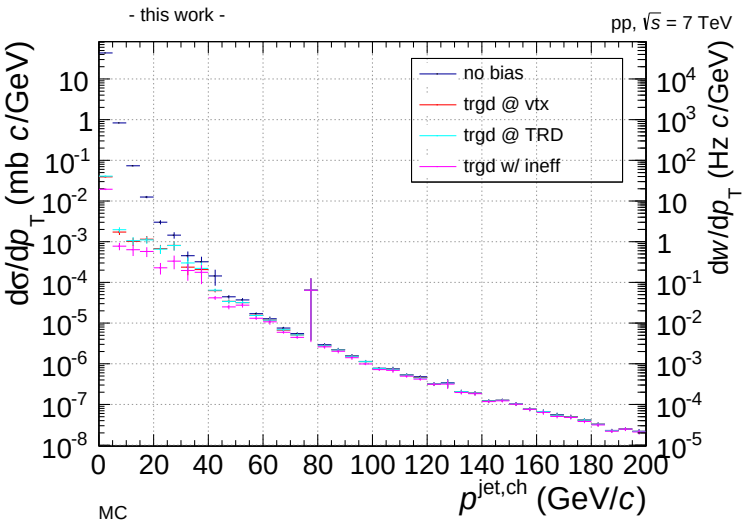
Plotting

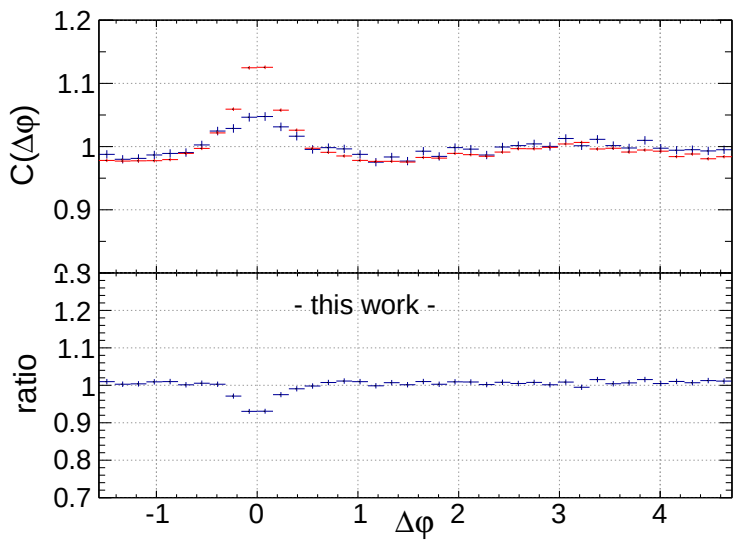
- plotting results is important part of producing scientific output
- often many (≥ 20) iterations (with final results),
  also as part of approval processes in the collaboration
  $\rightsquigarrow$ **very time-consuming** fine tuning of figures

criticism

- very hard to get consistent plotting behaviour, e.g. plots with multiple frames or axes always need hand-tuning for axes
- many standard tasks require manual treatment,
  e.g. statistical and systematic uncertainties, ratio MC/data, ...
- improved file format could help to adjust plots,
  e.g. contain values, settings, and figure
- alternative plotting packages could help
  (e.g. matplotlib, pgfplots, ...)

Plotting examples





- evolution from first look to "final" result (still subject to change)
- lengthy plotting macros (or plotting library) with lots of manual tweaking
- even more complicated with stat. & syst. uncertainties
- interactive approach does not help for tweaking
- would be really good to consider actual usage of plotting facilities by physicists

Merging

- memory resources are limited,
  typically $2 - 4$ GB per Grid job
- output from individual jobs needs to be merged:
  this often requires lots of memory, in particular for `THnSparse`
- merging particularly critical,
  shows up only at the very end
- typical failure of analysis trains:
  ending up with unmergeable output
- possible to reduce the memory footprint?

ALICE-specific work on memory-constrained merging on-going
[Marian Ivanov et al.]

C++ interpreter

- allows an easy start,
  also by allowing very sloppy code
- often paid later (poor code quality)
  when macros are converted to compiled code
- uncompiled macros in repositories often become obsolete
  since it goes unnoticed that they are broken by other modifications

---

significant disadvantages:
not a necessity but are "encouraged"

better separation of macros and compiled code,
or **at least stricter checks would be better**

Performance traps

- Clone() is extremely slow,
  but tempting to use $\Rightarrow$ add warning?
- THnSparse usage can be tempting in small-scale tests
  but not representative of final memory usage
  $\Rightarrow$ allow initialization with realistic memory usage?
  (e.g. if known from previous tests)
- automatic switching of sparse or full layout?
- for higher dimensional histograms:
  under-/overflow bins can take considerable space

Streamer limitations

- not all C++ constructs are supported
  (maybe changed in latest ROOT versions?)
- personal preference:
  leave C++ alone and decouple data storage (a la protobuf)
- schema evolution appears to have performance penalty
  but often not required, e.g.
  for online components or temporary buffering

ROOT as library

- monolithic design
- ROOT encourages to be used as main process
- often it would be nice to use ROOT libraries separately possible but lacks documentation and support
- better interfaces to other libraries

in particular file I/O

- lots of data available as ROOT files
- simple reading of data from ROOT files would be nice e.g. reading ROOT file on embedded system

ROOT trees

- standalone `TTree` often leads to problems for new users
- many different ways to initialize, e.g.
  Bronch vs Branch
- in ALICE: for standard users
  mostly hidden inside other classes

---

could probably be improved
by better documentation or clean up

Possible features for ROOT

- **TreeStream**
  - fill tree with cout like syntax
    but allowing for objects
  - hides all the initialization
    of variables and branches
  - easy to add for output/monitoring of intermediate results
- **StatToolkit**: collection of tools for
  - fast Gaussian fitting
  - linear *n*-dimensional fitting
  - easy plotting from THnSparse
  - ...

[Marian Ivanov]

Proposed features for ROOT

- **AliNDLocalRegression**:
  fit and store *n*-dimensional functions
  - functional and error representation
  - fitting
  - comparison of fit and data (QA)
  - persistence

- use cases:
  - calibration, e.g. parametrization of results
  - reconstruction, e.g. corrections
  - analysis, e.g. Monte-Carlo tuned on data
  - fast Monte-Carlo

[Marian Ivanov]

Wishlist/Ideas

- plotting/viewing on mobile devices
- memory information of ROOT objects, e.g. `THnSparse`, memory used for all objects in a `TList`

---

some are being worked on,
maybe even done?

Summary

- ROOT is our main software framework,
  and will continue to be so
- some avoidable limitations
- in some places we are struggling

Thank you

for your attention!