Some topics for discussion

31/03/2016

P. Hristov

"Wish list" for offline week (ALIROOT-6598)

- Thanks to Laurent for preparing this list:

1. Release validation :
   - what it is exactly ? how long does it takes ? can it be tailored for other purposes (e.g. validation the muon_calo passes ?)
2. Concerning the muon_calo passes.
   - Those has been acknowledged as regular passes for a while now, which is good. What's is less good is the state of the corresponding macros (for reconstruction, AOD filtering and QA) : what is the status here ? (e.g. can we have systematically something like a "is_muon_calo" to run only what's needed for those passes ?)
3. AliRoot tagging and deployment procedure.
   - I'm unclear of what's the procedure to get a fast deployment if ever needed (for instance for the Shuttle). Could that be clarified ?
4. Shuttle testing
   - I understand (and agree) that to get a change validated in the Pt2 Shuttle one has to pass the Shuttle Test first. For that one needs to commit to aliroot master. But during the time the test is performed if a tag is made (for whatever other reason) from the head of the master, then the still-in-testing-phase-shuttle-code can end up in the release, and might be used at Pt2 if the version used is changed there (I understand it's not automatic but it can happen).
5. c++11 allowed in AliRoot (HLT in particular) : when ?
6. ROOT 6 before Run3 ? (in particular vs an alice-specifc-root-5)

Release validation

- See the presentation of Dario
- My summary:
  - Different levels of validation (the longest is ~6h)
    - Compilation on different platforms
    - Technical tests (i.e. coverage analysis, profiling, Coverity)
    - Simulation/reconstruction/AOD using the AliRoot/test
    - CPass0/CPass1/PPass/AOD filtering of selected pp, pPb and PbPb runs
  - We plan to integrate everything relevant to the ALICE processing
    - SHUTTLE
    - Validation analysis train(s)
  - We would like to expose the results in a nice way
    - Automatic alarms
    - Trending plots
    - Root file
- We are open to proposals and contributions (i.e. test suits)

AliRoot tagging and deployment procedure.

- My summary
  - We profit from the validation cluster and after some initial checks tag release candidates (i.e. v5-08-03-rc3) from the master. Then we start the full validation chain;
  - In case of successful validation we convert the release candidate to release tag: v5-08-03-rc3 -> v5-08-03;
  - In case of unsuccessful validation we fix the issue and tag the next release candidate from the master. We start the release validation.
  - The process usually converges after 3-4 iterations;
  - When we have productions with a given set of tags and a problem is discovered, we branch from the relevant tag and cherry-pick the fix. This assumes the fix has no influence on the physics results;
  - If a serious problem is discovered in a production, we scratch it and use newer release to repeat it;
  - The deployment is done only for release tags, the release candidates are not exposed on CVMFS.

SHUTTLE testing

- My summary
  - So far we use dedicated machine to test the DA;
  - We plan to include it in the validation chain. Normally SHUTTLE will follow the release cycle of AliRoot;
  - SHUTTLE depends marginally on mainly on RAW data code, detector decoding, different maps, etc. and since these are relatively stable, may need its own release schedule;
  - In case of urgent needs for changes we can:
    - Cherry-pick in a branch from existing release tag
    - Make dedicated tag from the master

C++11 allowed in AliRoot (HLT in particular)

- The compilation with –std=c++11 is working and we are using it on a regular basis. The next step would be to add it “officially” to the options.
- The usage of C++11 extensions is not supported by CINT and Root5 I/O, so most probably they will not work in macros and for persistent data members.

ROOT 6 before Run3

- Our software has been working with Root6 since 2013
- My summary of Predrag's opinion ☺
  - Switching to ROOT 6 before we have consolidated our software and achieved desired functionality and precision (i.e. TPC reconstruction and calibration) would be too risky (in particular possible increase in memory requirements)
  - The practical benefit of switching to ROOT 6 is marginal (more precise error reporting and handling by the interpreter and possibility to use C++11)
  - Make sure that our software works with ROOT 6 so, once the preconditions in terms of s/w stability are met, we might switch to ROOT 6 but certainly not during current data taking period
  - For the moment we should use Root6 only in ALICE O2
- My proposal
  - Add Root6 validation and measure regularly its performance
  - Try Root6 in test GRID production/train to compare with Root5
  - We may soon get memory reduction wrt Root5 with Root6 modules