Subject: My analysis is slow
Posted by Bastian Löher on Thu, 13 Jun 2019 16:38:26 GMT
View Forum Message <> Reply to Message

This should initiate a discussion on how to cope with increasing demands on CPU power during analysis due to

- our increasing adoption of R3BRoot for all our detectors
- our increasing level of analysis (raw -> calib -> tracking)
- our increasing number of channels (100s -> 1000s)
- our increasing accepted trigger rate (1 kHz -> more than 30 kHz)

In the beginning it was barely noticeable that data analysis took time on the CPU, because

- few detectors were actually handled
- only a few tasks per detector were implemented
- incoming event rate was low

This year is a turning point with respect to most of these aspects, which lead to noticeable delays in the data analysis. Especially the online analysis was close to unusable at times during the s444/s473 data taking. This was partly amplified by the low network bandwidth and unlucky resource usage (analysis, data transport and user home directory on shared machine). In s454 the situation was a bit better, but mostly because major detector systems were not participating in the analysis (Neuland, AMS, PSP).

Also, when looking at the amount of data collected, we've reached a new record with stored lmd files of more than 40 TB. We've collected many millions of events, and offline data analysis can currently proceed only at a fraction of the acquisition speed. This means that analysing and re-analysing the full data sample takes up a considerable amount of time for the student / post-doc working with the data.

Therefore, I'd like to discuss the options we have for improving the performance of data analysis within R3BRoot and/or with the help of external tools.

A few ideas come to mind very quickly:

- parallelization using PROOF (old-fashioned?)
- fan-out of events from ucesb to several R3BRoot analysis processes, then merging histograms and trees in a final step (ucesb part is already implemented)
- parallelization using FairRoot framework (using FairMQ, use control macro to deploy to batch farm, what is the status there?)
- separate each FairTask into standalone 'micro-service', which always runs and processes data as soon as it is available (similar to DAQ nodes)

I believe we have to make a distinction here also for online and offline analysis, because different boundary conditions apply:

- online does not need full statistics, offline must process every event
- online precision can be lower (e.g. calibration parameters), offline should be as accurate as possible
- online should be single pass, offline can take multiple iterations
- online should result in histograms, offline should produce a tree for further processing

Please share your ideas, thoughts, suggestions, because this is the next important step we have to tackle regarding our data analysis.

Bastian
