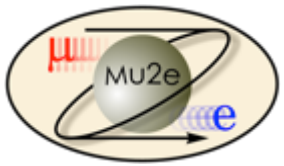
WP4
Mu2e Software updates

R.Donghia, LNF-INFN

MUSE Scientific Board Meeting
May-June 2019



MUSE

INFN
Istituto Nazionale di Fisica Nucleare

Outline

- Online Data monitoring
  - TDAQ: Good progress on all fronts
- Tracking optimization

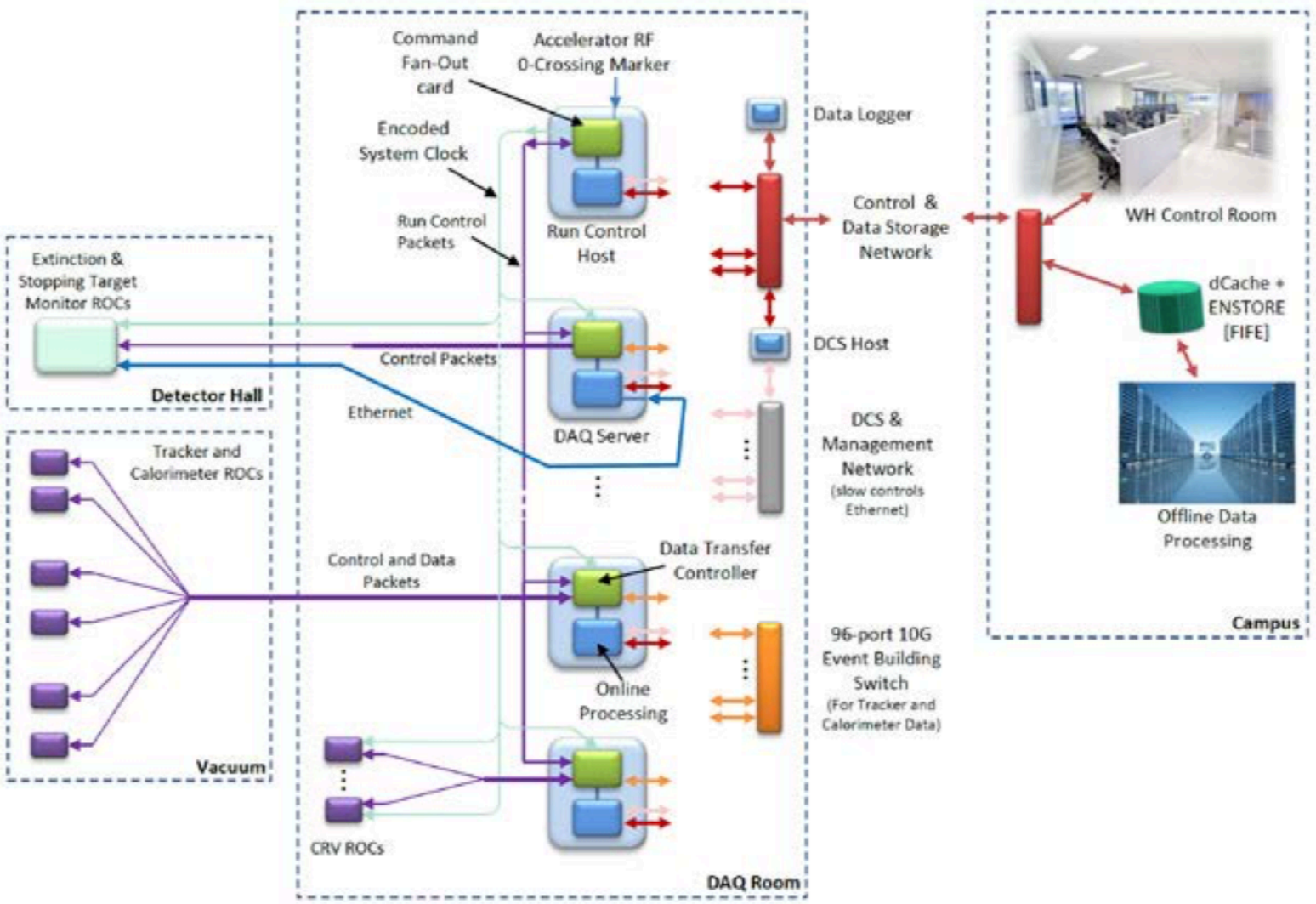
Data Monitoring Introduction

Trigger and DAQ system (TDAQ) must provide a set of monitoring tools to inspect the quality of the data collected by the detector. In addition, TDAQ must read and display information about the detector status from the EPICS server. Three different levels of data monitoring have been considered so far for Mu2e:

- online, real time data and slow control monitoring
- nearline, a limited look at the data with a latency of about 1 hour
- offline, full data reconstruction with a latency of a few hours to a day

DAQ system

Overview of the DAQ system together with the detector front-end electronics



Online monitoring

- Performed by sampling a fraction of the triggered data recorded by the data logger nodes using Dispatcher and Monitor *art* processes
- The persisted data can be kept for a pre-defined amount of time or stored on tape. The sampling fraction scales obviously with the number of Dispatcher/Monitor nodes
- Both the Dispatcher and Monitor are *art* processes, offering all the facilities of the *art* framework
- In this architecture, the Monitor nodes can only inspect triggered data
- While the Dispatcher / Monitoring processes and the trigger filter algorithms will run on the same hardware
  - During commissioning, the data rate should be low enough to dedicate more processing power to monitoring
  - During physics data taking, monitoring a small fraction of data
- The detector status will be monitored by the EPICS server. *Otsdaq* will request and display slow control data from EPICS.

Nearline Monitoring

- Limited look at all the data with a latency of ~1 hour as a validation before performing full reconstruction.
- While nearline monitoring would be desirable, we currently do **not** have the resources to perform it on the same hardware running the trigger filters. To mitigate the situation, we plan to reduce the latency to run the first pass of full reconstruction on the grid to the minimum
- Fast reconstruction algorithms (i.e. calorimeter cosmic reconstruction) could also be included in online monitoring depending on the performance cost

Offline Monitoring

- Once the data have been transferred to the 48-hour buffer disk, data monitoring becomes the responsibility of the computing group. Offline data monitoring and validation will be described in another document.

Tracking modules optimization

- The track trigger will run the first few modules of the tracking sequence (~3ms) – within the trigger budget, but a safety margin is prefered
- The bulk of the processing time in spent in the delta ray removal algorithm (mostly clustering algorithm) and the time cluster finder algorithm
- Look at potential optimizations for the FlagBkgHits and TimeClusterFinder modules that would only incur small loss of efficiency
- Few optimizations in ComboHitMaker and StrawHitMaker to speed up the code (no algorithmic change) done
- There are a few more ideas to implement that will yield small improvement, but significant gains would likely need a dedicated specialist (FPGA algorithm or substantial modifications to framework).

Tracking modules Summary

- Worked on FlagBkgHits and TimeClusterFinder modules to (i) simplify code, (ii) improve timing and (iii) investigate potential improvements in efficiency.
- Improved timing performance of FlagBkgHits by 30% (0.3 ms) and TimeClusterFinder by 75% (0.5 ms) for a relative loss in efficiency less than 2% - this could be made even smaller once the module parameters are optimized
- Extrapolating from the latest trigger processing time benchmarks, the track trigger should run in ~3 ms with 20 threads running simultaneously (might be interesting to look at cache usage).
- At this point, decompression and hit creation start accounting for about 50% of the time. Not sure how much better we can do!

Next steps:

- updated this code to the latest Offline version
- Implement algorithm improvement in TimeClusterFinder and continue tracking inefficiencies