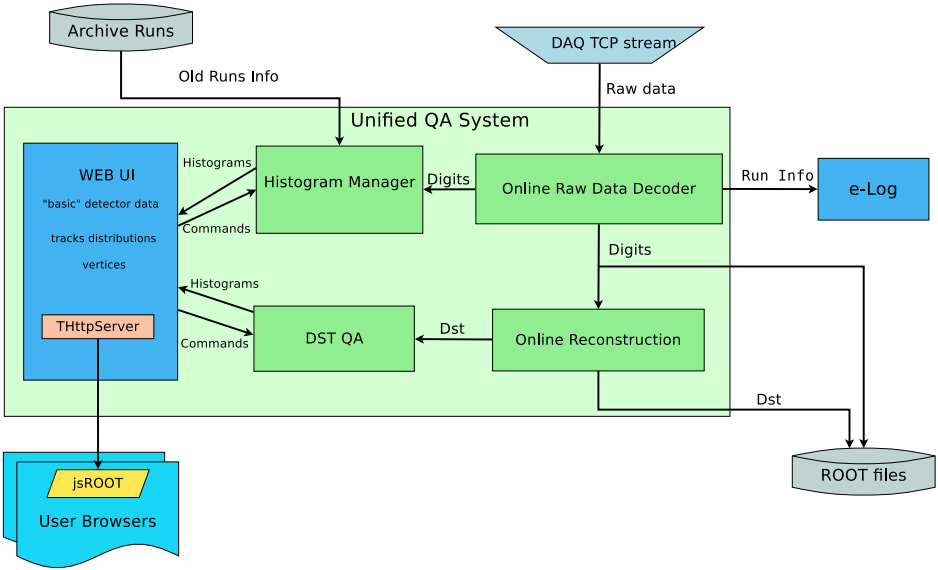
Status of the Raw Data Converter for Run 8 data processing

Ilnur Gabdrakhmanov

Joint Institute for Nuclear Research, Laboratory of High Energy Physics

Analysis & Software Meeting of the BM@N Experiment
September 13, 2023

General system scheme

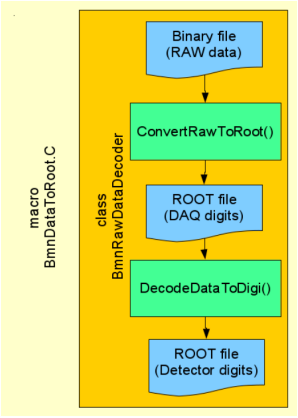


My zone of responsibility - by gray color

Detector	Decoding	Reco/HitMaker	Monitoring: Digi	DST
Triggers	✓	-	✓	-
Silicon	✓	✓	✓	✓
SiBT	✓	✓	✓	✓
GEM	✓	✓	✓	✓
CSC	✓	✓	✓	✓
DCH	✓	✓	✓	✗
ToF400	✓	✓	✓	✓
ToF700	✓	✓	✓ (needs extension)	✓
ScWall	✓	✓	✓	✓
FHCal	✓	✓	✓	✓
Hodoscope	✓	✓	✓	✓
NDet	✓	✓	✓	✓
Profilometer	✓ ✗(needs reworking)	-	✓ (needs refactoring)	-
GlobalTracking	-	✓	-	✓

- Thanks to the calorimeter group which developed their part of decoding & monitoring.
- Profilometer part implemented in a separate pipeline separated from the main DAQ.

Decoding scheme



First step (Data Converter):

- Read a binary data file with RAW-data.
- Parse the data blocks: run/spill/event/module.
- Create «DAQ-digits» (ADC, TDC, TQDC, HRB, TTVXS, etc.) accordingly DAQ-data-format and write them into a tree.

Second step (Data Decoder):

- Read detector mappings (channel-to-strip) from the Unified Database
- Calculate pedestals and common modes of channels
- Clear noisy channels
- Decode DAQ-digits into detector-digits (BmnGemDigit, BmnTofDigit, etc.)
- Write the tree with detector-digits to a ROOT-file

Converter/Decoder updates

- General:
  - Class BmnRawDataDecoder disentangled into two separate classes for converting and decoding respectively
  - JSON config data blocks are processed
  - The corresponding DAQ trigger configuration is written into DigiRunHeader
  - So as trigger channel configuration (before/after protection, reduction factors)
- Raw data converter (by A. Islentev):
  - Converting procedure has been parallelized
  - The conversion speed scales well, the bottleneck is a writing to the TTree from the threads
- Decoder (by A. Islentev):
  - New noisy channel detection algorithm for the forward silicon detector
  - Several fixes for the denoising in strip detectors

Get Run Header

$$DigiRunHeader * runHeader = (DigiRunHeader*) file->Get("RawRunHeader");$$

DAQ trgger configurations

$$BmnTrigConfig\& trigConfig = runHeader->GetTrigConfig(); // \text{Get trig configuration}$$

```
trigConfig.GetMatchWindow()     // Match window
trigConfig.GetCalibCount()      // Count of calibration events
trigConfig.GetCalibEn()         // Are calibration events enabled
trigConfig.TrigChannelConfig()  // Settings for each trigger channel
```

Trigger channel config

$$map\langle string, BmnTrigChannelConfig\rangle\& channelConfig = trigConfig.TrigChannelConfig(); // \text{Get trig channel config}$$

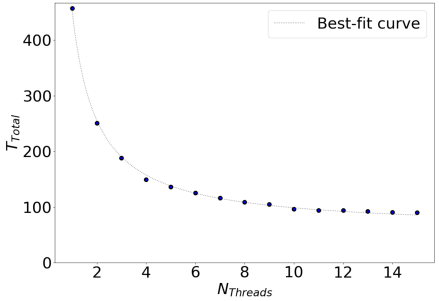
```
channelConfig["CCT1"].after_protect_en      // After protection enabled
channelConfig["CCT1"].after_protect_time    // After protection time
channelConfig["CCT1"].before_protect_en     // Before protection enabled
channelConfig["CCT1"].before_protect_time   // Before protection time
channelConfig["CCT1"].delay                 // Delay
channelConfig["CCT1"].reduction_en          // Is reduction enabled
channelConfig["CCT1"].reduction_factor      // Reduction factor
```

Terminal example

```
Attaching file bmn_run7282_ev0_p0_digi.root as _file0...
(TFile *) 0x562c023b9af0
root [1] DigiRunHeader* r = (DigiRunHeader*)_file0->Get("DigiRunHeader;")
(DigiRunHeader *) 0x562c048a6c40
root [2] BmnTrigConfig tc = r->GetTrigConfig()
(BmnTrigConfig &) Name: Title:
root [3] tc.GetMatchWindow()
(unsigned int) 40
root [4] tc.GetCalibCount()
(unsigned int) 100
root [5] tc.TrigChannelConfig()
(std::map<std::string, BmnTrigChannelConfig> &) { "BC1L" => @0x562c08244ef0, "BT" => @0x562c081d8cc0, "BT&nBUSY" => @0x562c081d55b0, "CCT1" => @0x562c082476a0, "CCT2" => @0x562c081a7c60,
 "MBT" => @0x562c081dbe40, "NiT" => @0x562c0277c740, "pBT" => @0x562c08119240 }
root [6] tc.TrigChannelConfig()["BT"].reduction_en
(bool) true
root [7] tc.TrigChannelConfig()["BT"].reduction_factor
(unsigned int) 2999
root [8] tc.TrigChannelConfig()["CCT1"].after_protect_en
(bool) false
root [9] tc.TrigChannelConfig()["BT"].after_protect_time
(unsigned int) 50
root [10] tc.TrigChannelConfig()["BT"].print()
 after_prot en    0
 after_prot  t   50
before_prot en    0
before_prot  t  100
        delay    0
  reduction en    1
reduct  factor 2999
root [11]
```

Figure: Getting trigger config

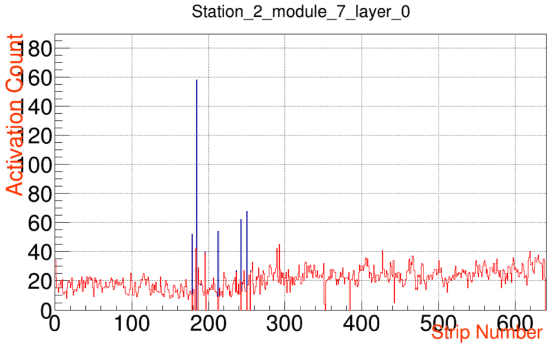
Converter parallelization (by A. Islentev)



Raw data converter parallelization:

- Working class - BmnConverterThread
- The main thread reads raw event blocks and passes them to working threads
- The "worker" threads parse events and write root branches to the trees
- After converting the block of events is being reordered and saved into separate files by spill
- For mass production of separate run files it is less efficient terms of overall [time x processors] expenses
- But for the online processing it will be a huge advantage

Forward silicon decoding improvements (by A. Islentev)



FSD noise filtering improvements:

- Noise detection by sliding averages implemented (instead of fixed blocks of strips)
- Fixes for the forward silicon detectors concerning their constructive details

Combined effect of improvements on DST level:

- In terms of amount of reconstructed events and PV width the effect is small
- Further tests of particles' yields and background levels are needed

BM@N

Conclusion

Since last collaboration meeting:

- General Converter/Decoder updates:
  - Converter/decoder separation
  - Extraction of trigger setting from embedded JSON blocks and storing them into the digi header
  - Converter parallelization
- Algorithmic improvements for the strip decoder:
  - New FSD denoising algorithm
  - Several important fixes

Work to do (concerning converting/decoding)

- Further tests of effects on DST are needed
- integrate parallel converting with the online QA

Thanks for your attention!