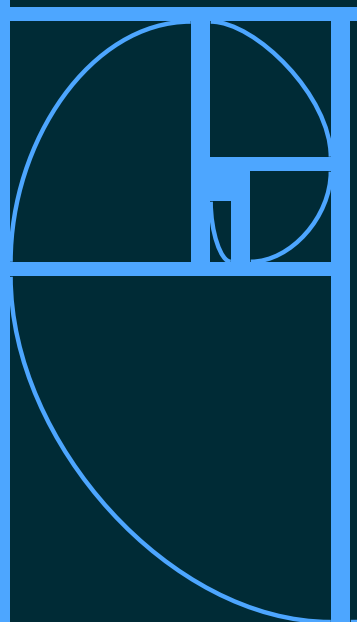
Trigger: v5 opmon

DS / PP meeting
September 24, 2024



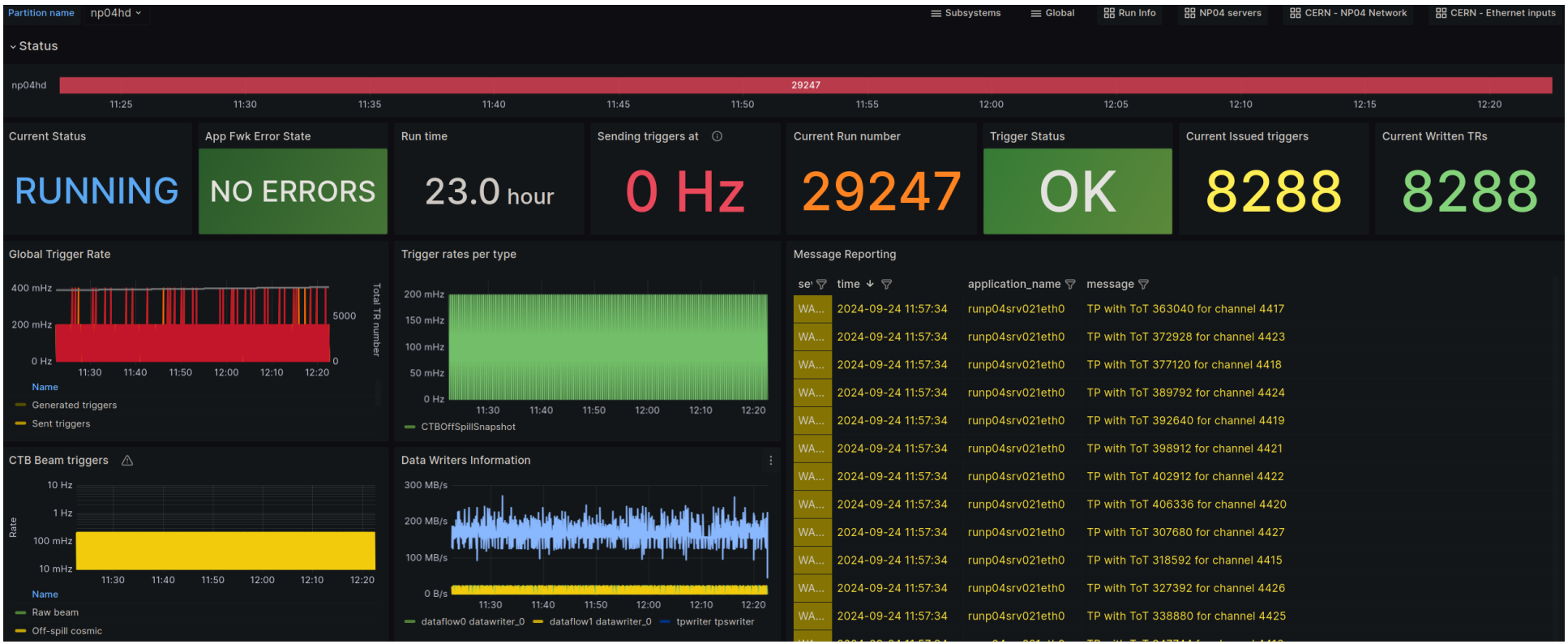
Michal Rigan
mrigan@sussex.ac.uk

University of Sussex

Opmon

- Opmon = operational monitoring
- dunedaq modules publish data to DB (json)
- DB is queried by grafana -> nice plots
- very important during real data-taking (our window to what is happening)
- up to developers of each team to implement what/when is being published

Grafana


Partition name
np04hd
Subsystems
Global
Run Info
NP04 servers
CERN - NP04 Network
CERN - Ethernet inputs
Status
np04hd
29247
11:25
11:30
11:35
11:40
11:45
11:50
11:55
12:00
12:05
12:10
12:15
12:20
Current Status
RUNNING
App Fwk Error State
NO ERRORS
Run time
23.0 hour
Sending triggers at
0 Hz
Current Run number
29247
Trigger Status
OK
Current Issued triggers
8288
Current Written TRs
8288
Global Trigger Rate
400 mHz
200 mHz
0 Hz
5000
0
Total TR number
Name
Generated triggers
Sent triggers
Trigger rates per type
200 mHz
150 mHz
100 mHz
50 mHz
0 Hz
CTBOffSpillSnapshot
Message Reporting
time
application_name
message
WA...
2024-09-24 11:57:34
runp04srv021eth0
TP with ToT 363040 for channel 4417
TP with ToT 372928 for channel 4423
TP with ToT 377120 for channel 4418
TP with ToT 389792 for channel 4424
TP with ToT 392640 for channel 4419
TP with ToT 398912 for channel 4421
TP with ToT 402912 for channel 4422
TP with ToT 406336 for channel 4420
TP with ToT 307680 for channel 4427
TP with ToT 318592 for channel 4415
TP with ToT 327392 for channel 4426
TP with ToT 338880 for channel 4425
CTB Beam triggers
10 Hz
1 Hz
100 mHz
10 mHz
Rate
Name
Raw beam
Off-spill cosmic
Data Writers Information
300 MB/s
200 MB/s
100 MB/s
0 B/s
dataflow0 datawriter_0
dataflow1 datawriter_0
tpwriter tpswriter

Trigger v5

- no opmon was ported when moving v4 -> v5
- new opmon system -> moving to protobuf files (Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data)

```
syntax = "proto3";

package dunedaq.trigger.opmon;

// Message representing TP Processor Information
message TPProcessorInfo {
  uint32 tp_received_count = 1;    // Number of TPs received
  uint32 ta_made_count = 2;        // Number of TAs made
  uint32 ta_sent_count = 3;        // Number of TAs sent
  uint32 ta_failed_sent_count = 4; // Number of TAs that failed to be sent
}
```

Trigger v5: opmon flow

1. opmon message

```
syntax = "proto3";

package dunedaq.trigger.opmon;

// Message representing TP Processor Information
message TPProcessorInfo {
  uint32 tp_received_count = 1;    // Number of TPs received
  uint32 ta_made_count = 2;        // Number of TAs made
  uint32 ta_sent_count = 3;        // Number of TAs sent
  uint32 ta_failed_sent_count = 4; // Number of TAs that failed to be sent
}
```

2. Define variables

```
std::atomic<uint64_t> m_tp_received_count{ 0 };  /
std::atomic<uint64_t> m_ta_made_count{ 0 };
std::atomic<uint64_t> m_ta_sent_count{ 0 };
std::atomic<uint64_t> m_ta_failed_sent_count{ 0 };
```

3. Actual logic to update

```
while (tas.size()) {
    m_ta_made_count++;
    if (!m_ta_sink->try_send(std::move(tas.back()), iomanager::Sender::s_no_block)) {
      ers::warning(TADropped(ERS_HERE, tp->tp.time_start, m_sourceid.id));
      m_ta_failed_sent_count++;
    } else {
      m_ta_sent_count++;
    }
    tas.pop_back();
}
return;
```

4. Publish

```
void
TPProcessor::generate_opmon_data()
{
  opmon::TPProcessorInfo info;

  info.set_tp_received_count( m_tp_received_count.load() );
  info.set_ta_made_count( m_ta_made_count.load() );
  info.set_ta_sent_count( m_ta_sent_count.load() );
  info.set_ta_failed_sent_count( m_ta_failed_sent_count.load() );

  this->publish(std::move(info));
}
```

Trigger v5: opmon flow

5. Published message


```
{
 "time": "2024-09-24T10:42:39.123582658Z",
 "origin": {
  "session": "michal-session",
  "application": "ru-01",
  "substructure": [
   "tphandler-1000",
   "TPProcessor",
   "tp-processor"
  ]
 },
 "measurement": "dunedaq.trigger.opmon.TPProcessorInfo",
 "data": {
  "ta_failed_sent_count": {
   "uint4_value": 0
  },
  "ta_sent_count": {
   "uint4_value": 898
  },
  "ta_made_count": {
   "uint4_value": 898
  },
  "tp_received_count": {
   "uint4_value": 898056
  }
 }
}
```


6. Grafana view

Trigger v5: structure

- Now using processors ('instead' of makers): TPProcessor, TAProcessor, TCProcessor
- (v5) MLT functionality also separated into TCProcessor and MLT:
  - TCProcessor: receives TCs, creates TDs. Deals with bitword checks, ignoring, merging. Passes on TD to MLT.
  - MLT: receives TDs, forwards actual readout requests. Interfaces with DFO (status) and Livetime Counter.
- plugins: RandomTCM, CustomTCM, TriggerPrimitiveMaker (replay), HSI Source Model

Trigger v5: opmon implementation

- most variables are incrementing, cumulative counters
- Grafana setup to show the difference (derivates)
- some weird cases (below)

Trigger v5: opmon implementation: simple

```
syntax = "proto3";

package dunedaq.trigger.opmon;

// Message representing TP Processor Information
message TPProcessorInfo {
  uint32 tp_received_count = 1;    // Number of TPs received
  uint32 ta_made_count = 2;        // Number of TAs made
  uint32 ta_sent_count = 3;        // Number of TAs sent
  uint32 ta_failed_sent_count = 4; // Number of TAs that failed to be sent
}
```

```
syntax = "proto3";

package dunedaq.trigger.opmon;

// Message representing HSI Source Model Information
message HSISourceModelInfo {
  uint32 received_events_count = 1; // Number of received HSI signals
  uint32 tcs_made_count = 2;        // Number of TCs made
  uint32 tcs_sent_count = 3;        // Number of TCs succesfully sent
  uint32 tcs_dropped_count = 4;     // Number of TCs dropped (failed to send)
}
```

Trigger v5: opmon implementation: complex

```
1   syntax = "proto3";
2
3   package dunedaq.trigger.opmon;
4
5   // Message representing Trigger Candidate Processor Information
6   message TCProcessorInfo {
7     uint32 tds_created_count = 1;           // Number of created TDs (requests)
8     uint32 tds_sent_count = 2;              // Number of successfully sent TDs (requests)
9     uint32 tds_dropped_count = 3;           // Number of dropped (failed to send) TDs (requests)
10    uint32 tds_failed_bitword_count = 4;    // Number of TDs (requests) that failed bitword check
11    uint32 tds_cleared_count = 5;           // Number of TDs (requests) cleared from pending at run status change
12    uint32 tc_received_count = 10;          // Number of received TCs
13    uint32 tc_ignored_count = 11;           // Number of ignored TCs
14    uint32 tds_created_tc_count = 21;       // Number of TCs contributing to created TDs (requests)
15    uint32 tds_sent_tc_count = 22;          // Number of TCs contributing to send TDs (requests)
16    uint32 tds_dropped_tc_count = 23;       // Number of TCs contributing to dropped TDs (requests)
17    uint32 tds_failed_bitword_tc_count = 24; // Number of TCs contributing to TDs (requests) that failed the bitword check
18    uint32 tds_cleared_tc_count = 25;       // Number of TCs contributing to TDs (requests) that were cleared at run stage change
19  }
```

Trigger v5: opmon implementation: complex

```
// Message representing Module Level Trigger Information
message ModuleLevelTriggerInfo {
  uint32 td_msg_received_count = 1;                // Number of trigger decision messages received at MLT from TCProcessor
  uint32 td_sent_count = 2;                        // Number of trigger decisions added to queue
  uint32 td_inhibited_count = 3;                   // Number of trigger decisions inhibited
  uint32 td_paused_count = 4;                      // Number of trigger decisions created during pause mode
  uint32 td_queue_timeout_expired_err_count = 5;   // Number of trigger decisions failed to be added to queue due to timeout
  uint32 td_total_count = 6;                       // Total number of trigger decisions created
  uint32 lc_klive = 10;                            // Total time [ms] spent in Live state - alive to triggers
  uint32 lc_kpaused = 11;                          // Total time [ms] spent in Paused state - paused to triggers
  uint32 lc_kdead = 12;                            // Total time [ms] spent in Dead state - dead to triggers
}

// Message representing TD Information, these counters are published separately for each TC type
// states >1 are mutually exclusive states
message TriggerDecisionInfo {
  uint32 received = 1;        // Number of received
  uint32 sent = 2;            // Number of sent
  uint32 failed_send = 3;     // Number of failed to send (network issue)
  uint32 paused = 4;          // Number of paused (triggers are paused)
  uint32 inhibited = 5;       // Number of inhibited (DFO is busy)
}
```

Trigger v5: opmon implementation: summary logs

```
MLT opmon counters summary:
-----------------------------
Received TD messages:        164
Sent TDs:                    138
Inhibited TDs:               0
Paused TDs:                  26
Queue timeout TDs:           0
Total TDs:                   164
-----------------------------
Livetime::Live:      69643
Livetime::Paused:    15999
Livetime::Dead:      0
```

Trigger v5: latency opmon

- in review, Grafana plots ready
- new latency class, used throughout:
  - compares data times to system time (std::chrono::system_clock)
- can deal with *ms*, μs
- shared protobuf msg
- additional case to deal with TD request readout windows
- configurable, added to appmodel schemas
- can sometimes be 'weird', as the reporting is not tracing objects uniquely (ie, a TA will report it's time_start when made)
- using 'offline' system/session there can be patterns observed that are not expected for online system (development)

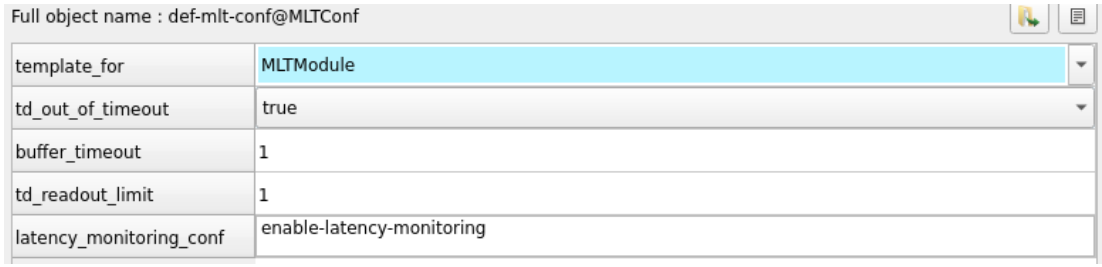
```
18    class Latency {
21      // Enumeration for selecting time units
22      enum class TimeUnit { Milliseconds, Microseconds };
23
24      // Constructor with optional time unit selection (defaults to Milliseconds)
25      Latency(TimeUnit time_unit = TimeUnit::Milliseconds)
26        : m_latency_in(0), m_latency_out(0), m_time_unit(time_unit)
27      {
28        // Set the clock tick conversion factor based on time unit
29        if (m_time_unit == TimeUnit::Milliseconds) {
30          m_clock_ticks_conversion = 16 * 1e-6; // For milliseconds: 1 tick = 16 * 10^-6 ms
31        } else {
32          m_clock_ticks_conversion = 16 * 1e-3;
33        }
34        // to convert 62.5MHz clock ticks to ms: 1/62500000 = 0.000000016 <- seconds per tick; 0.000016 <- ms per tick;
35        // 16*1e-6 <- sci notation
36      }
37
38      // Function to get the current system time in ms or ns based on time unit
39      uint64_t get_current_system_time() const ...
48      }
49
50      // Function to update latency_in
51      void update_latency_in(uint64_t latency)
52      {
53        m_latency_in.store(latency * m_clock_ticks_conversion);
54      }
55
56      // Function to update latency_out
57      void update_latency_out(uint64_t latency)
58      {
59        m_latency_out.store(latency * m_clock_ticks_conversion);
60      }
61
62      // Function to get the value of latency_in
63      uint64_t get_latency_in() const ...
73      }
74
75      // Function to get the value of latency_out
76      uint64_t get_latency_out() const ...
84      }
85
86    private:
87      std::atomic<uint64_t> m_latency_in;  // Member variable to store latency_in
88      std::atomic<uint64_t> m_latency_out; // Member variable to store latency_out
89      double m_clock_ticks_conversion; // Dynamically adjusted conversion factor for clock ticks
90      TimeUnit m_time_unit;  // Member variable to store the selected time unit (ms or ns)
```

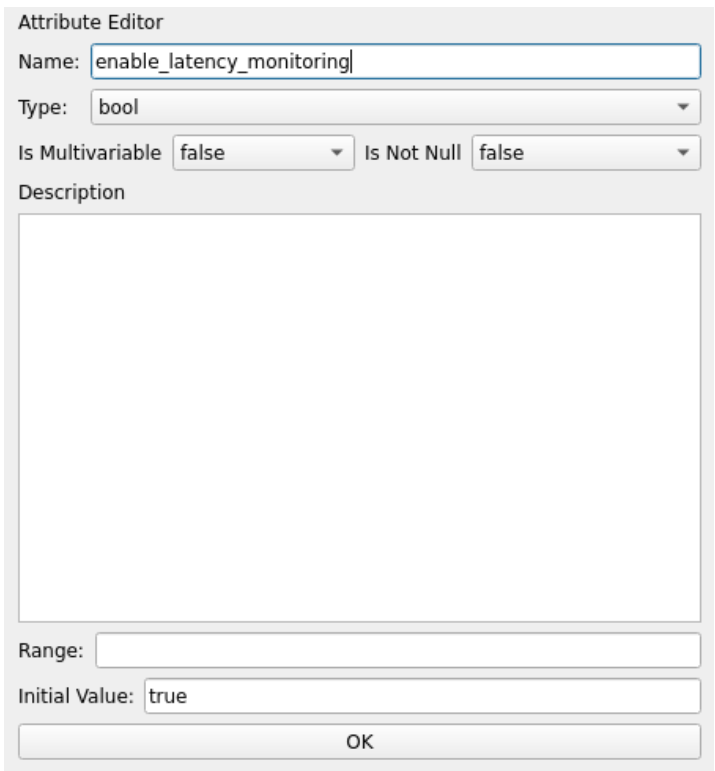
Trigger v5: latency opmon: proto

```
1   syntax = "proto3";
2
3   package dunedaq.trigger.opmon;
4
5   // Message for latency variables
6   // Latency represents the difference between current system (clock) time and the data time of particular (TX) data object
7   // Units are ms
8   // Used by many trigger modules
9   message TriggerLatency {
10    uint32 latency_in = 1;
11    uint32 latency_out = 2;
12  }
```

```
28  // Message for MLT TD requests latency vars
29  // Latency represents the difference between current system (clock) time and the requested TD readout window (start/end)
30  // Units are currently us (but use an enum and can be changed)
31  message ModuleLevelTriggerRequestLatency {
32    uint32 latency_window_start = 1;
33    uint32 latency_window_end = 2;
34  }
```

Trigger v5: latency opmon: oks





Gragana

(now for the fun part)