Upstream DAQ Readout software integration to appfwk

Roland Sipos
CERN

DUNE DAQ/SC Meeting
8th October 2020

Outline of the talk

- Upstream DAQ Readout system
- ProtoDUNE versions
- OnHost BR
- Modularization
- Integration to appfwk
- Outlook and Summary

The goal of this talk

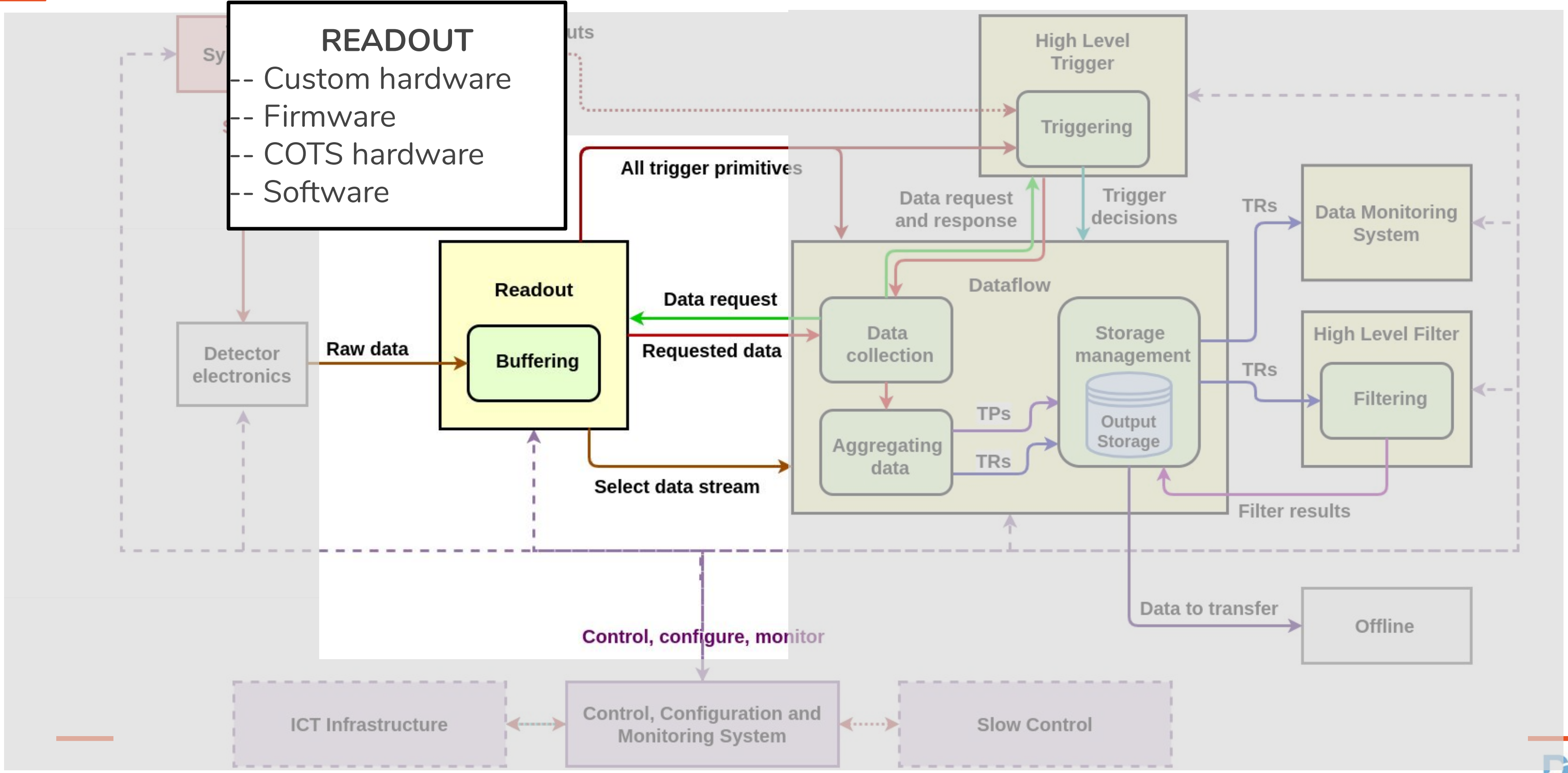
Based on the experience with the FELIX readout in ProtoDUNE-SP, I would like to make some remarks on the existing system's problems and what lead to those.

Then I will present the new approach in order to try solving those.

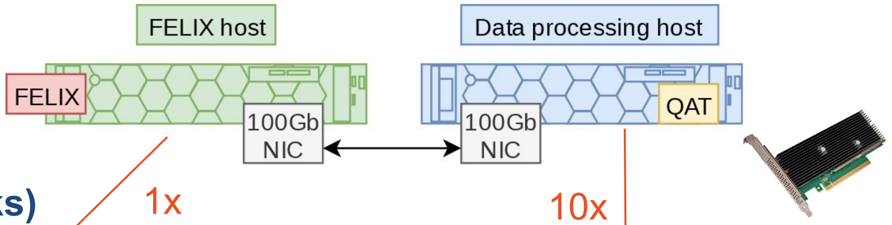
Finally, the first experiences of the integration of the readout DAQModules to the DUNE DAQ appfwk.

Upstream DAQ

Readout System functions

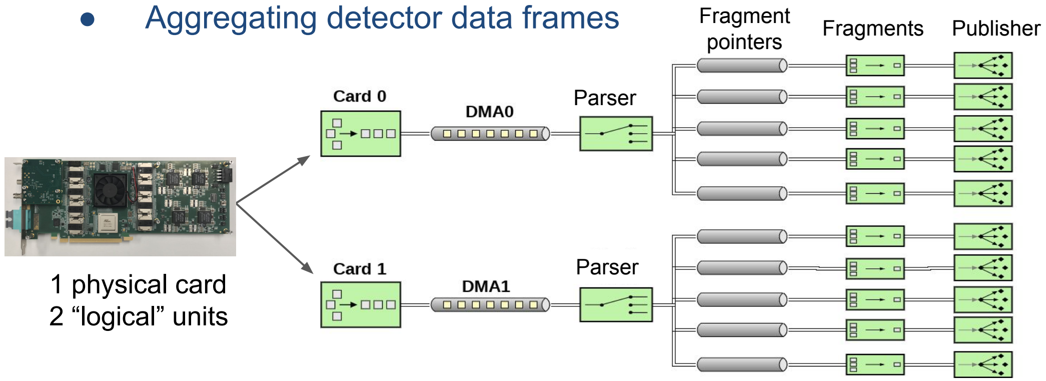

READOUT
-- Custom hardware
-- Firmware
-- COTS hardware
-- Software

FELIX in ProtoDUNE



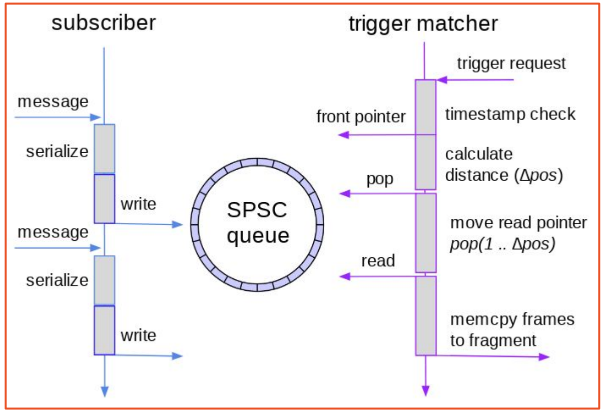
1 FELIX card handles a single APA (10x ~1GB/s links)

Modest modifications on FPGA gateware
for lower memory I/O rate:
- Increasing DMA payload size
- Aggregating detector data frames

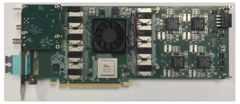


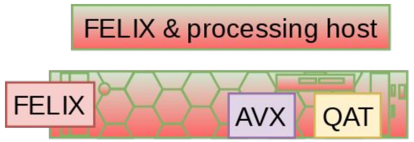
Data routing software is customized:
- Scatter-gather: collects pointers for detector data fragments
- Single copy pipeline: serialization to user buffer
- Data published on Infiniband over Ethernet
- **Felix-star uses a similar principle!**

BoardReader process
- Subscribes to single link & buffers data
- Extracts data fragments for triggers
- Reorders data (AVX2 & 512)
- Hardware accelerated compression
  - Intel® QuickAssist (QAT)



OnHost BoardReader



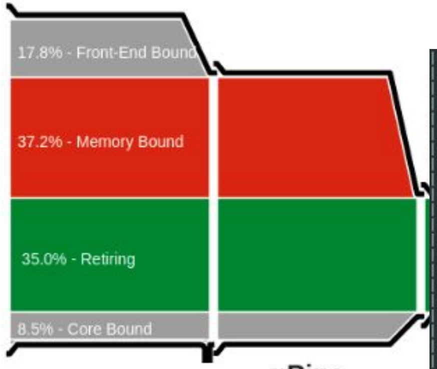


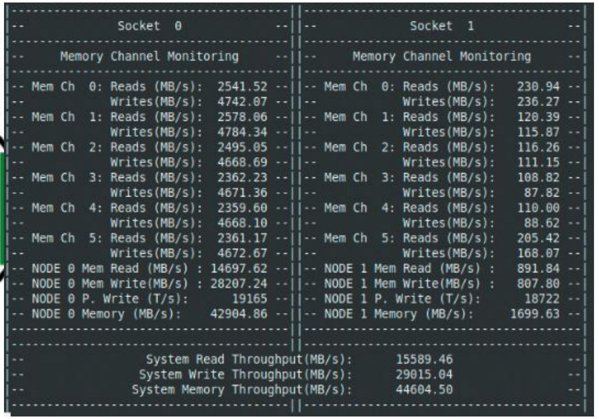
Merged FELIX data routing software, with data selection (trigger-matching) algorithm

- Eliminated 100Gb P2P connection
- Stuffed every existing RO feature to a single ArtDAQ BoardReader
- Extensive server evaluation
  - PCIe riser configurations,
  - BIOS settings
- Optimized for memory throughput
  - Performance profiling
- Heavy NUMA balancing between processing threads and allocated memory
- Interrupt moderation of 10Gb NIC

2 x OnHost process

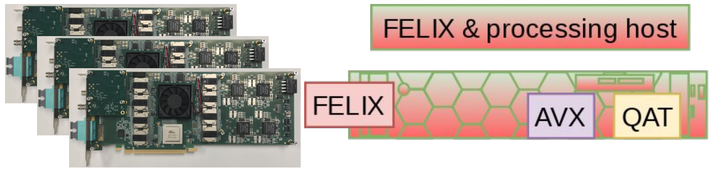
1 x DMA parser
5 x link parser → High memory I/O
5 x hit finder → High CPU util.
5 x trigger matcher



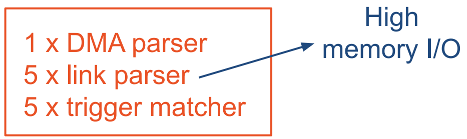


```
|---------------------------------------||---------------------------------------|
|--             Socket  0             --||--             Socket  1             --|
|---------------------------------------||---------------------------------------|
|--     Memory Channel Monitoring     --||--     Memory Channel Monitoring     --|
|---------------------------------------||---------------------------------------|
|-- Mem Ch  0: Reads (MB/s):  2541.52 --||-- Mem Ch  0: Reads (MB/s):   230.94 --|
|--            Writes(MB/s):  4742.07 --||--            Writes(MB/s):   236.27 --|
|-- Mem Ch  1: Reads (MB/s):  2578.06 --||-- Mem Ch  1: Reads (MB/s):   120.39 --|
|--            Writes(MB/s):  4784.34 --||--            Writes(MB/s):   115.87 --|
|-- Mem Ch  2: Reads (MB/s):  2495.05 --||-- Mem Ch  2: Reads (MB/s):   116.26 --|
|--            Writes(MB/s):  4668.69 --||--            Writes(MB/s):   111.15 --|
|-- Mem Ch  3: Reads (MB/s):  2362.23 --||-- Mem Ch  3: Reads (MB/s):   108.82 --|
|--            Writes(MB/s):  4671.36 --||--            Writes(MB/s):    87.82 --|
|-- Mem Ch  4: Reads (MB/s):  2359.60 --||-- Mem Ch  4: Reads (MB/s):   110.00 --|
|--            Writes(MB/s):  4668.10 --||--            Writes(MB/s):    88.62 --|
|-- Mem Ch  5: Reads (MB/s):  2361.17 --||-- Mem Ch  5: Reads (MB/s):   205.42 --|
|--            Writes(MB/s):  4672.67 --||--            Writes(MB/s):   168.07 --|
|-- NODE 0 Mem Read (MB/s) : 14697.62 --||-- NODE 1 Mem Read (MB/s) :   891.84 --|
|-- NODE 0 Mem Write(MB/s) : 28207.24 --||-- NODE 1 Mem Write(MB/s) :   807.80 --|
|-- NODE 0 P. Write (T/s):      19165 --||-- NODE 1 P. Write (T/s):      18722 --|
|-- NODE 0 Memory (MB/s):    42904.86 --||-- NODE 1 Memory (MB/s):     1699.63 --|
|---------------------------------------||---------------------------------------|
|---------------------------------------||---------------------------------------|
|--                 System Read Throughput(MB/s):      15589.46                --|
|--                System Write Throughput(MB/s):      29015.04                --|
|--               System Memory Throughput(MB/s):      44604.50                --|
|---------------------------------------||---------------------------------------|
```

OnHost BoardReader 2



6 x OnHost process



Locality and performance characteristics understood

- 2 x APAs - demonstrated within ProtoDUNE run environment
- One full socket available for post-processing

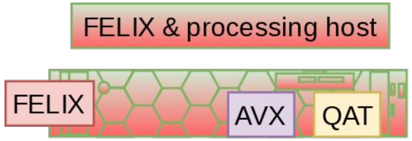
Other studies also done:

- 3 x APAs - emulated data with fake triggering demonstrated on the same system

The solution scales on a well understood manner

The talk does not focus on performance indicators, but purely about implementation shortcomings in the FELIX OnHost Boardreader.

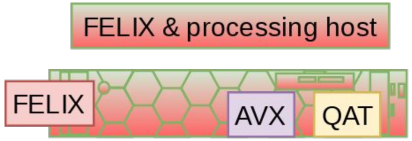
OnHost BoardReader - Problems



Extremely monolithic approach
The result became the anti-pattern: “big ball of mud”

- Backward compatibility attempt and bloated feature set
  - P2P mode, frame reordering, QAT compression
  - Linked against unnecessary and heavy dependencies (libqat, ibverbs, etc.)
  - Forces platform dependent build (avx intrinsics, QAT)
  - No interfaces straightforward interfaces for TP handling
- Hardwired logic based on configuration combinations with override
  - E.g.: If (mode == “OnHost” && num_sources == 2) // this must be 10 links! <- bad
  - Hardcoded constants and cryptic identifiers
- Tree hierarchy in resource management
  - Hierarchy between different resources and processing element are not clear
  - Central “glue” holds every member and prepare resources for them
- Custom and unconventional ways in some components
  - Avoided framework features (tracing, logging, statistics handling)
- Complete lack of documentation

OnHost BoardReader - What to do?



Cleanup or structural redesign and reimplementation?

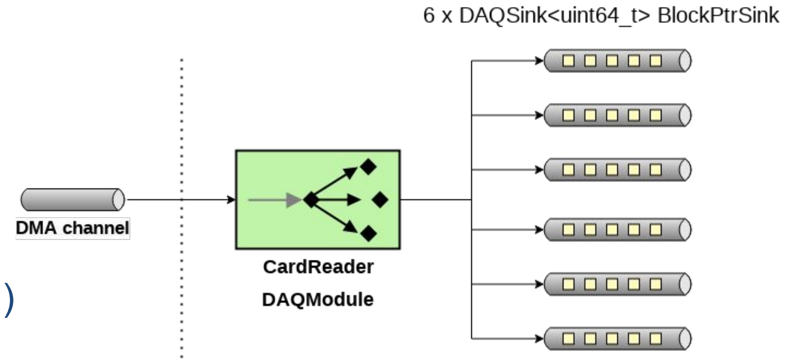
- Fearing from the cleanup resulting in “yet another OhBR version”, I decided to make an attempt to re-design the Readout software, following the principles of the appfwk

1. Identify resources
   a. Physical FELIX Card, Memory that holds WIB frames, Sockets that publish Hits
2. Identify processing elements
   a. DMA Parser, Block/Link Parser, TriggerMatcher, RequestReceiver
3. Identify connections between the resources and processing elements
   a. DMA Parser reads card, produces pointers
   b. Parsers consume pointers and produce WIB frames
   c. TriggerMatchers use WIB frame containers and publish on their custom network layer
4. Group the components into a processing pipeline

CardReaderModule

Reads the FELIX card’s DMA channel and based on the Block header, it queues in its pointer to the corresponding DAQ sink

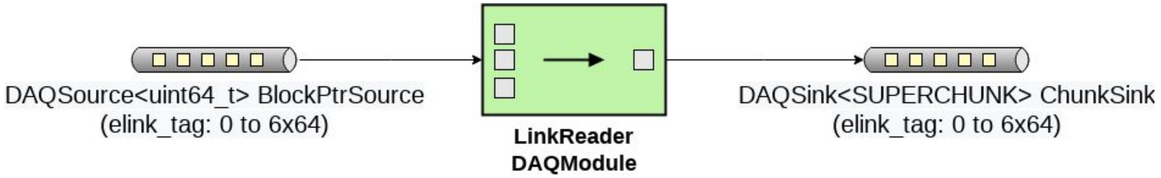
- Resources
  - FLX DMA buffer (populated by every enabled logical link’s output)
- Processing stage
  - Communicates with the card’s driver to advance the read pointers, indicating that we have “seen” the new block
  - The pointer to the “seen” block is stored in the link’s specific queue
- Sinks
  - Queues for pointers (for links that are enabled!)
- Remarks:
  - Only this module needs to be linked against libFlxCard.so from the FELIX dependencies



LinkReaderModule

Parses FELIX SuperChunks from a single BlockPtrSource resource

- Resources
  - BlockPtrSource of link that the Module need to process
- Processing stage
  - Merge the fragments of the chunks that span across multiple blocks
  - Report problems on data corruption
  - Push chunks to WIB frame buffer
- Sinks
  - Latency buffers that hold the actual raw data

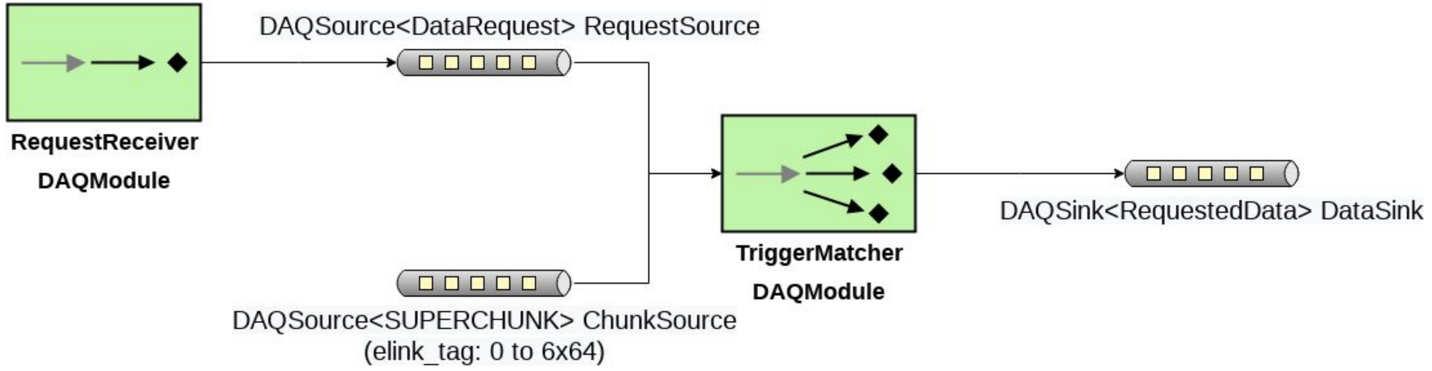

DAQSource<uint64_t> BlockPtrSource
(elink_tag: 0 to 6x64)
LinkReader
DAQModule
DAQSink<SUPERCHUNK> ChunkSink
(elink_tag: 0 to 6x64)

Should it be available to read from multiple BlockPtrSource and spawn processing units in the same Module?
Sometimes an extra scaling capability may compromise clarity…

TriggerMatcherModule

Finds requested data in the latency buffers

- Resources
  - Circular buffer of raw WIB binary data
- Processing stage
  - Extracts data from buffers based on the requests
- Sinks
  - Requested data sinks

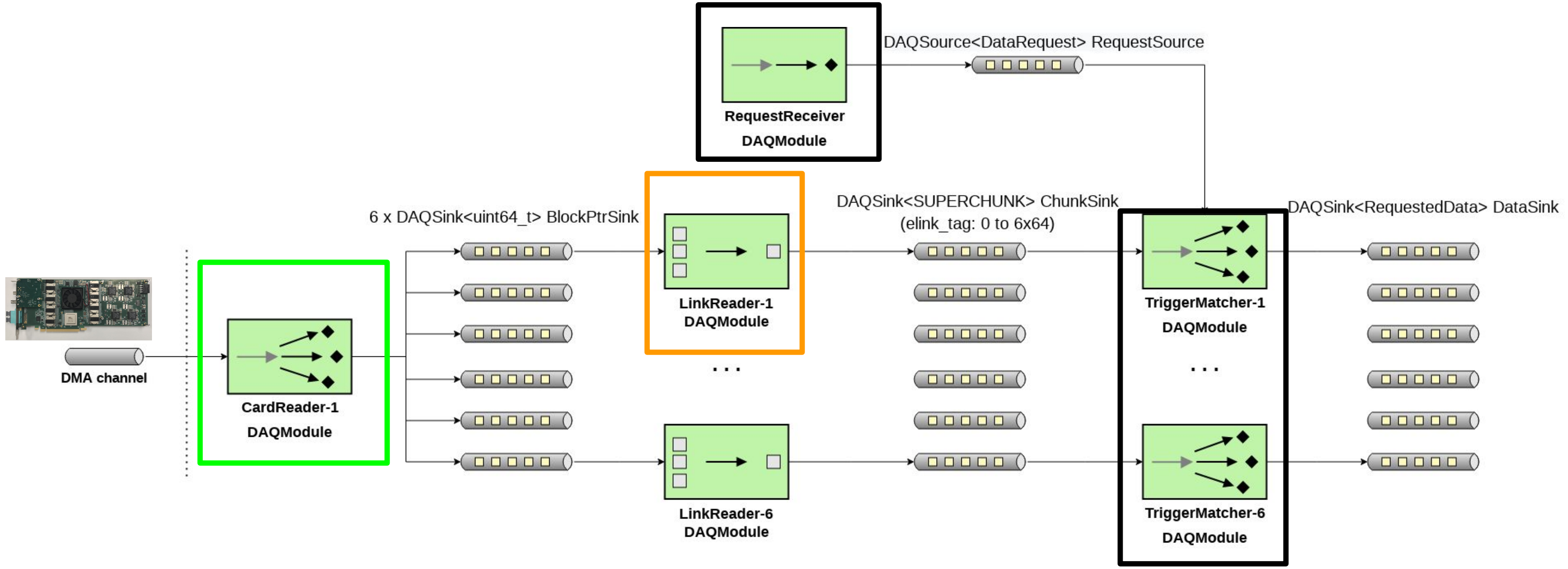

DAQSource<DataRequest> RequestSource
RequestReceiver
DAQModule
TriggerMatcher
DAQModule
DAQSink<RequestedData> DataSink
DAQSource<SUPERCHUNK> ChunkSource
(elink_tag: 0 to 6x64)

Other features/components

- SNB Buffer needs some experimentation how to connect in the the topology
  - E.g.: specific DataRequest, streamed by the TriggerMatcher to dedicated, special sinks
- DataRequest handling need to be extended with specific RequestTypes
  - Might need to drive the TriggerMatching mechanism based on that
- TriggerMatcher redesign
  - Need to support overlapping triggers and variable request window sizes
  - Direction is clear, easier solution exist than expected
  - Non-intrusive to existing sink/source types
- Fake mode
  - I want to extend the LinkReader with a “fake mode” that is able to replay raw data files over-and-over and re-create O(10s) latency buffer with content in memory.
  - Add fake-trigger to TriggerMatcher: extract last available data with random window (ms-s)
  - Add fake-trigger to request mode
- Stretch goals
  - Compression DAQmodule using QuickAssist (Compress data from inputs to outputs)
  - Post-processor DAQModule (Async callback wrapper to do “whatever with extracted data)

Integration

Missing modules are coming at a steady pace...
https://github.com/roland-sipos/udaq-readout.git



Summary & Outlook

Summary:

- All in all, really good experience so far
  - Fought with some build issues on the beginning. Needed to get familiar with the framework a bit
  - Great support and guidance from the developers! Thanks a lot for it.

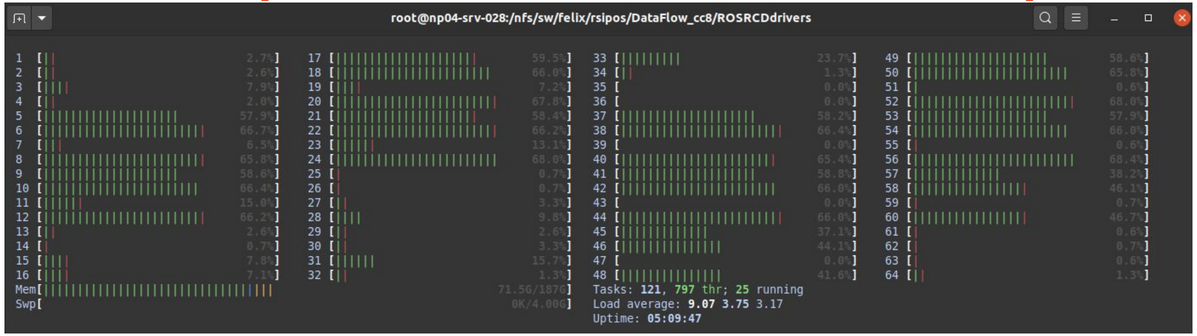
Outlook:

- Ex. CardReader I want every Module to operate in emulation mode, providing a full readout layer for testing
- Provide fake-mode chain ASAP for Dataflow development (HDF5 Module)
- Provide full-chain (with CardReader) ASAP for MiniDAQ

End

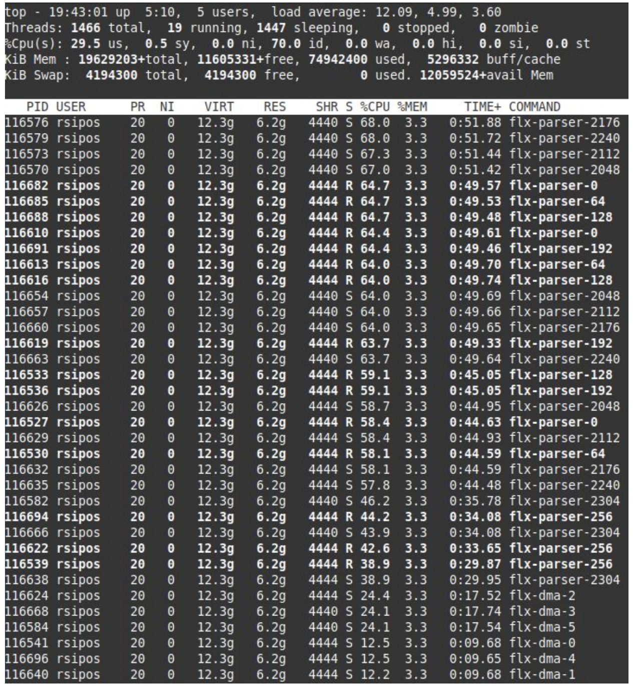
- Feedback is more than welcome!

Backup - Readout software performance



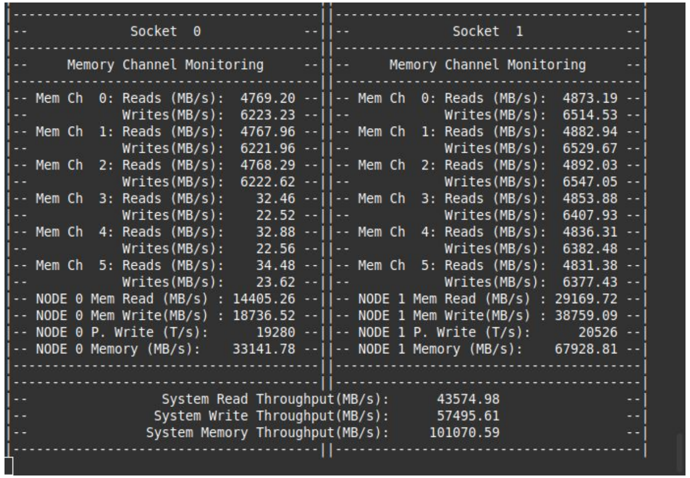
root@np04-srv-028:/nfs/sw/felix/rsipos/DataFlow_cc8/ROSRCDdrivers

```
  1  [||                         2.7%]   17 [|||||||||||||||||||||     59.5%]   33 [|||||||||                 23.7%]   49 [||||||||||||||||||||      58.6%]
  2  [||                         2.6%]   18 [|||||||||||||||||||||||   66.0%]   34 [||                         1.3%]   50 [|||||||||||||||||||||||   65.8%]
  3  [||||                       7.9%]   19 [||||                       7.2%]   35 [                           0.0%]   51 [|                          0.6%]
  4  [||                         2.0%]   20 [||||||||||||||||||||||||  67.8%]   36 [                           0.0%]   52 [||||||||||||||||||||||||  68.0%]
  5  [||||||||||||||||||||      57.9%]   21 [|||||||||||||||||||||     58.4%]   37 [||||||||||||||||||||      58.2%]   53 [||||||||||||||||||||      57.9%]
  6  [||||||||||||||||||||||||  66.7%]   22 [||||||||||||||||||||||||  66.2%]   38 [||||||||||||||||||||||||  66.4%]   54 [|||||||||||||||||||||||   66.0%]
  7  [|||                        6.5%]   23 [||||||                    13.1%]   39 [                           0.0%]   55 [|                          0.6%]
  8  [||||||||||||||||||||||||  65.8%]   24 [||||||||||||||||||||||||  68.0%]   40 [|||||||||||||||||||||||   65.4%]   56 [||||||||||||||||||||||||  68.4%]
  9  [||||||||||||||||||||      58.6%]   25 [|                          0.7%]   41 [||||||||||||||||||||      58.8%]   57 [|||||||||||||             38.2%]
 10  [|||||||||||||||||||||||   66.4%]   26 [|                          0.7%]   42 [|||||||||||||||||||||||   66.0%]   58 [|||||||||||||||||         46.1%]
 11  [||||||                    15.0%]   27 [||                         3.3%]   43 [                           0.0%]   59 [|                          0.7%]
 12  [||||||||||||||||||||||||  66.2%]   28 [||||                       9.8%]   44 [||||||||||||||||||||||||  66.0%]   60 [|||||||||||||||||         46.7%]
 13  [||                         2.6%]   29 [||                         2.6%]   45 [|||||||||||||             37.1%]   61 [|                          0.6%]
 14  [|                          0.7%]   30 [||                         3.3%]   46 [|||||||||||||||           44.1%]   62 [|                          0.7%]
 15  [||||                       7.8%]   31 [||||||                    15.7%]   47 [                           0.0%]   63 [|                          0.6%]
 16  [||||                       7.1%]   32 [||                         1.3%]   48 [|||||||||||||||           41.6%]   64 [||                         1.3%]
 Mem[||||||||||||||||||||||||||||||||||                 71.5G/187G]   Tasks: 121, 797 thr; 25 running
 Swp[                                                      0K/4.00G]   Load average: 9.07 3.75 3.17
                                                                       Uptime: 05:09:47
```



```
top - 19:43:01 up  5:10,  5 users,  load average: 12.09, 4.99, 3.60
Threads: 1466 total,  19 running, 1447 sleeping,   0 stopped,   0 zombie
%Cpu(s): 29.5 us,  0.5 sy,  0.0 ni, 70.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 19629203+total, 11605331+free, 74942400 used,  5296332 buff/cache
KiB Swap:  4194300 total,  4194300 free,        0 used. 12059524+avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
116576 rsipos    20   0   12.3g   6.2g   4440 S  68.0  3.3   0:51.88 flx-parser-2176
116579 rsipos    20   0   12.3g   6.2g   4440 S  68.0  3.3   0:51.72 flx-parser-2240
116573 rsipos    20   0   12.3g   6.2g   4440 S  67.3  3.3   0:51.44 flx-parser-2112
116570 rsipos    20   0   12.3g   6.2g   4440 S  67.0  3.3   0:51.42 flx-parser-2048
116682 rsipos    20   0   12.3g   6.2g   4444 R  64.7  3.3   0:49.57 flx-parser-0
116685 rsipos    20   0   12.3g   6.2g   4444 R  64.7  3.3   0:49.53 flx-parser-64
116688 rsipos    20   0   12.3g   6.2g   4444 R  64.7  3.3   0:49.48 flx-parser-128
116610 rsipos    20   0   12.3g   6.2g   4444 R  64.4  3.3   0:49.61 flx-parser-0
116691 rsipos    20   0   12.3g   6.2g   4444 R  64.4  3.3   0:49.46 flx-parser-192
116613 rsipos    20   0   12.3g   6.2g   4444 R  64.0  3.3   0:49.70 flx-parser-64
116616 rsipos    20   0   12.3g   6.2g   4444 R  64.0  3.3   0:49.74 flx-parser-128
116654 rsipos    20   0   12.3g   6.2g   4440 S  64.0  3.3   0:49.69 flx-parser-2048
116657 rsipos    20   0   12.3g   6.2g   4440 S  64.0  3.3   0:49.66 flx-parser-2112
116660 rsipos    20   0   12.3g   6.2g   4440 S  64.0  3.3   0:49.65 flx-parser-2176
116619 rsipos    20   0   12.3g   6.2g   4444 R  63.7  3.3   0:49.33 flx-parser-192
116663 rsipos    20   0   12.3g   6.2g   4440 S  63.7  3.3   0:49.64 flx-parser-2240
116533 rsipos    20   0   12.3g   6.2g   4444 R  59.1  3.3   0:45.05 flx-parser-128
116536 rsipos    20   0   12.3g   6.2g   4444 R  59.1  3.3   0:45.05 flx-parser-192
116626 rsipos    20   0   12.3g   6.2g   4444 S  58.7  3.3   0:44.95 flx-parser-2048
116527 rsipos    20   0   12.3g   6.2g   4444 R  58.4  3.3   0:44.63 flx-parser-0
116629 rsipos    20   0   12.3g   6.2g   4444 S  58.4  3.3   0:44.93 flx-parser-2112
116530 rsipos    20   0   12.3g   6.2g   4444 R  58.1  3.3   0:44.59 flx-parser-64
116632 rsipos    20   0   12.3g   6.2g   4444 S  58.1  3.3   0:44.59 flx-parser-2176
116635 rsipos    20   0   12.3g   6.2g   4444 S  57.8  3.3   0:44.48 flx-parser-2240
116582 rsipos    20   0   12.3g   6.2g   4440 S  46.2  3.3   0:35.78 flx-parser-2304
116694 rsipos    20   0   12.3g   6.2g   4444 R  44.2  3.3   0:34.08 flx-parser-256
116666 rsipos    20   0   12.3g   6.2g   4440 S  43.9  3.3   0:34.08 flx-parser-2304
116622 rsipos    20   0   12.3g   6.2g   4444 R  42.6  3.3   0:33.65 flx-parser-256
116539 rsipos    20   0   12.3g   6.2g   4444 R  38.9  3.3   0:29.87 flx-parser-256
116638 rsipos    20   0   12.3g   6.2g   4444 S  38.9  3.3   0:29.95 flx-parser-2304
116624 rsipos    20   0   12.3g   6.2g   4444 S  24.4  3.3   0:17.52 flx-dma-2
116668 rsipos    20   0   12.3g   6.2g   4440 S  24.1  3.3   0:17.74 flx-dma-3
116584 rsipos    20   0   12.3g   6.2g   4440 S  24.1  3.3   0:17.54 flx-dma-5
116541 rsipos    20   0   12.3g   6.2g   4444 S  12.5  3.3   0:09.68 flx-dma-0
116696 rsipos    20   0   12.3g   6.2g   4444 S  12.5  3.3   0:09.65 flx-dma-4
116640 rsipos    20   0   12.3g   6.2g   4444 S  12.2  3.3   0:09.68 flx-dma-1
```



```
|---------------------------------------||---------------------------------------|
|--             Socket  0             --||--             Socket  1             --|
|---------------------------------------||---------------------------------------|
|--     Memory Channel Monitoring     --||--     Memory Channel Monitoring     --|
|---------------------------------------||---------------------------------------|
|-- Mem Ch  0: Reads (MB/s):   4769.20 --||-- Mem Ch  0: Reads (MB/s):   4873.19 --|
|--            Writes(MB/s):   6223.23 --||--            Writes(MB/s):   6514.53 --|
|-- Mem Ch  1: Reads (MB/s):   4767.96 --||-- Mem Ch  1: Reads (MB/s):   4882.94 --|
|--            Writes(MB/s):   6221.96 --||--            Writes(MB/s):   6529.67 --|
|-- Mem Ch  2: Reads (MB/s):   4768.29 --||-- Mem Ch  2: Reads (MB/s):   4892.03 --|
|--            Writes(MB/s):   6222.62 --||--            Writes(MB/s):   6547.05 --|
|-- Mem Ch  3: Reads (MB/s):     32.46 --||-- Mem Ch  3: Reads (MB/s):   4853.88 --|
|--            Writes(MB/s):     22.52 --||--            Writes(MB/s):   6407.93 --|
|-- Mem Ch  4: Reads (MB/s):     32.88 --||-- Mem Ch  4: Reads (MB/s):   4836.31 --|
|--            Writes(MB/s):     22.56 --||--            Writes(MB/s):   6382.48 --|
|-- Mem Ch  5: Reads (MB/s):     34.48 --||-- Mem Ch  5: Reads (MB/s):   4831.38 --|
|--            Writes(MB/s):     23.62 --||--            Writes(MB/s):   6377.43 --|
|-- NODE 0 Mem Read (MB/s) :  14405.26 --||-- NODE 1 Mem Read (MB/s) :  29169.72 --|
|-- NODE 0 Mem Write(MB/s) :  18736.52 --||-- NODE 1 Mem Write(MB/s) :  38759.09 --|
|-- NODE 0 P. Write (T/s):       19280 --||-- NODE 1 P. Write (T/s):       20526 --|
|-- NODE 0 Memory (MB/s):     33141.78 --||-- NODE 1 Memory (MB/s):     67928.81 --|
|---------------------------------------||---------------------------------------|
|---------------------------------------||---------------------------------------|
|--               System Read Throughput(MB/s):      43574.98                  --|
|--              System Write Throughput(MB/s):      57495.61                  --|
|--             System Memory Throughput(MB/s):     101070.59                  --|
|---------------------------------------||---------------------------------------|
```