A horizontally scalable online processing system for trigger-less data acquisition

Matteo Migliorini, Jacopo Pazzini, Andrea Triossi, Marco Zanetti
University and INFN Padova
9 march 2022
Triggering in HEP experiments is a key aspect

- At LHC, most of the collision events are “uninteresting” physics
- Saving everything on disk would be challenging
- **Online Triggers** allow to reduce the rate and select interesting events
- “Necessary evil” for most experiments
  - Discarded events are lost forever
  - Limit the type of processing you can do

Selection of 1 event in 10,000,000,000,000
Introduction: trigger-less readout

Continuous trigger-less readout systems provide an alternative to increase the efficiency and acceptance under a number of experimental conditions

- Read all the outputs from the detector and perform an online analysis
- This comes with the price of designing custom online data processing schemas with strong emphasis on specific requirements

However, a number of applications might profit from a trigger-less readout and data processing system where performance can be sacrificed to gain generality and portability

- Testbeams and small-scale experiments
  - Often triggerless readout is mandatory
- Alternative/complementary DAQ “branches” for large-scale experiments
  - Check where the standard trigger is inefficient
Introduction: our work

What could be done to generalize the streaming readout processing with currently available computing frameworks?

We propose an Online Stream Processing Pipeline based on distributed computing frameworks

- Rely on publicly available frameworks for the development
- Bring tools used for offline batch processing to the “online” world

Focus on scalability and portability

- Scalable to account for various degrees of throughput with a single solution
- Portable to be able to deploy the processing solution on multiple computing infrastructures

Developed using a small-scale experimental setup used as a testbed for development
Test setup @LNL (Padova)

- Muon telescope with 4 “miniDTs”
  - Reduced area replica of CMS’ DTs (muon detector in CMS’ barrel)
- Signals produced by the chamber collected and digitized by two FPGAs (VC707)
- Hits transmitted as soon as available to a backend board using GBT protocol
Sector test @CMS

- Local testbed excellent small-scale mockup of Phase2 CMS DT detector up to the front end devices

- One sector of CMS DTs equipped with the prototypes of the new FE boards

- Possibility to test our work on a large scale experiment

- CMS is investigating the possibility to devote a branch of the DAQ for triggerless data acquisition and processing for Phase2 (40MHz scouting project)
  - Trigger primitives read before trigger decision
Stream from the FE collected on a backend board mounted on the PCIe of a commercial server

- Data stream processed by an **online reconstruction algorithm**
  - Based on neural networks on FPGA
- Results of the processing inserted back into the original stream of hits
- Stream into a buffer before performing the **DMA transfer** to the server memory
  - Create batches of hits and segments
- Each batch is reformatted and sent to a kafka cluster
  - During development, a copy of the hits is saved on the local disk for offline evaluation
Online segment reconstruction on FPGA

- Backend board acting as a data concentrator for multiple input links (up to 8 per board)

- Algorithm based on neural network to identify muon track segment online
  - Hits and segments are merged into a single data stream
  - NO data filtering or suppression
Distributed processing framework

- Multiple backend servers, each reading a subset of FE boards
- Asynchronous readout
  - BE servers have different delays
  - Hits and segments are not time-ordered
- Kafka used as a buffer to decouple readout and processing
Stream structure

- Hits encode “raw” quantities
  - Information about where and when hit has been produced

- Hits produced by the detector have a fixed format
  - Structured stream

- View it as a DataFrame

- Columnar-wise operations on subset of hits to compute relevant quantities

- Hits distributed on multiple nodes

<table>
<thead>
<tr>
<th>HEAD</th>
<th>FPGA</th>
<th>TDC_CHANNEL</th>
<th>ORBIT_CNT</th>
<th>BX_COUNTER</th>
<th>TDC_MEAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>25</td>
<td>5899610</td>
<td>2930</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>25</td>
<td>5899618</td>
<td>309</td>
<td>19.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>42</td>
<td>5899621</td>
<td>1157</td>
<td>26.0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>75</td>
<td>5899646</td>
<td>2500</td>
<td>29.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>124</td>
<td>5899658</td>
<td>3151</td>
<td>7.0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>111</td>
<td>5899658</td>
<td>3153</td>
<td>26.0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>42</td>
<td>5899658</td>
<td>3150</td>
<td>3.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>122</td>
<td>5899658</td>
<td>3155</td>
<td>17.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>56</td>
<td>5899658</td>
<td>3153</td>
<td>4.0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>43</td>
<td>5899658</td>
<td>3157</td>
<td>27.0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>113</td>
<td>5899658</td>
<td>3152</td>
<td>3.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>123</td>
<td>5899658</td>
<td>3155</td>
<td>8.0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>114</td>
<td>5899658</td>
<td>3157</td>
<td>17.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>57</td>
<td>5899658</td>
<td>3153</td>
<td>19.0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>41</td>
<td>5899658</td>
<td>3161</td>
<td>11.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>125</td>
<td>5899658</td>
<td>3163</td>
<td>25.0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>138</td>
<td>5899658</td>
<td>3163</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>55</td>
<td>5899658</td>
<td>3155</td>
<td>29.0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>44</td>
<td>5899658</td>
<td>3160</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>112</td>
<td>5899658</td>
<td>3160</td>
<td>12.0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>9</td>
<td>5899658</td>
<td>3148</td>
<td>15.0</td>
</tr>
</tbody>
</table>

Reconstruction algo. hit
Distributed processing framework

Use a cluster computing framework such as Spark and Dask

- Both offers the concept of distributed dataframe and are widely used for offline analytics
Distributed processing

There are multiple quantities that one may be interested in extracting from the stream

- **Low-level processing**
  - Data aggregations used to produce low-level features (occupancy, rate, ...)
  - Use them as Data Quality Monitoring (DQM) of the detector using all the incoming data
    - Not biased by a trigger selection

- **Mid-level processing**
  - Combine all the segments produced by the same muon on different stations
  - Compute higher-level structures, e.g. events, for later processing stages

- **High-level processing**
  - Analysis level algorithms running on the results of the intermediate processing stages
  - ML applications for complex tasks such as anomaly detection
Stream processing

- Each worker reads from a single kafka partition
  - Kafka partitions mapped into dask dataframe partitions
- Stream divided into batches of 5 seconds
  - One dataframe created for each batch
- Results written into two topics for further processing/visualization
- Horizontally scalable
  - Add brokers/nodes to increment the processing throughput
Kubernetes implementation

- All components are containerized and orchestrated with Kubernetes
  - Easy to scale up/down based on the needs
  - Portable by construction

- Processed data are saved to an object storage accessible via S3 interface

- WebUI developed using plotly
  - DQM plots
  - Controls for action and configurations
  - Services communicates using dedicated kafka topics
Performance: backend server

- Throughput of the reco. algorithm fixed by design
- Benchmark DMA transfer and Kafka producer throughput

- Performance of the DMA transfer from the FPGA to the host limited by the PCIe gen3 throughput
- Well above the expected throughput of the detector

- Performance of the kafka producer measured using the standard kafka-producer-perf-test
  - Cluster with 3 brokers, 30 partitions topic
- Measured similar performance using librdkafka
- Parameters tuned to maximize throughput
  - Message size: 2Kb
  - Batch size: 200k
  - Compression: lz4
- Measured ~1.3 GB/s
Performance: dask

Initial test performed on a small cluster hosted on cloud veneto infrastructure (Padova University)

- 5 virtual machine with 4 vCPU and 16GB of RAM each

Performance depends on the type of processing

- Currently measured only for low level processing
- Unpack input stream + compute monitoring quantities

Observed limitations

- Implemented in python, heavy use of numpy
- However, some steps forced to use python
  - E.g. moving data from kafka to numpy
- Bad memory management, memory not being released
What’s next?

Working to replicate the setup at CERN

- 1 backend server installed at Point 5 and correctly reading data from a front end board
- At the end of this month we will install a second server and fibers to read the entire sector
- Expected ~400 MB/s from one sector
  - With our setup it is possible to measure this number, up to now only estimated
  - Know the real noise ratio
- First CMS’ sub-detector read triggerless at FE level!

Move processing on GPUs

- Use RAPIDS, library developed by NVIDIA to perform data analytics on GPUs
- Won hardware grant, one A100 available
- Should be well integrated with both Spark and Dask
Outlook

- Testing and implementing an online processing pipeline for triggerless data
- Focus on portability of the system
  - System can be easily deployed using kubernetes
- Use publicly available tools and frameworks
  - Data analytics frameworks widely used in big-data applications

Future work:

- Install the readout system at CMS
- Test the processing pipeline at CERN
  - It would be great to replicate the cloud environment used in Padova
  - In the future a fraction of CMS’ resources will be available
  - Test setup in the DAQ lab @building 40: send test data from there to CERN cloud?
- Explore RAPIDS
  - Promising results from the first tests