Triggerless Data Acquisition System for the AMBER Experiment

**Introduction**

* AMBER (Apparatus for Meson and Baryon Experimental Research) is a fixed target experiment that is located at the M2 beam line of the CERN Super Proton Synchrotron (SPS)
* It is the successor of the COMPASS experiment, which aims to study the formation of hadrons by quarks and gluons, as well as their distinctive properties. Some of the findings of the COMPASS experiment include those relating to the spin structure of the proton, as well as the pion’s polarizability (i.e. the degree to which the pion’s constituent positive and negative electric charges can be separated in an electric field)
* The pilot run took place in 2021, with the next test run scheduled for this year
* The initial aim of the free-running DAQ is to measure the proton radius on an active hydrogen time projection chamber (TPC) with a muon beam

**Structure of Experiment**

* TPC filled with up to 20 bars of hydrogen (2000 kpa) is used as a target onto which muons are incident
* Muon scattering angle is determined using four tracking detectors (fast detectors)
* Slow detectors (e.g. TPC) have a very long drift time (120 microsconds)
* Merging the slow detectors with the fast ones enables a continuous stream of data (streaming DAQ)

**Triggerless Data Acquisition**

* Continuous readout of detectors
* HLT is used instead of a low-level trigger logic
* General reduction scheme (i.e. any detector can participate in the filter). Both hardware and software processing are employed
* Goal is 10 GB/s. High performance software and hardware is required to achieve this

**Readout Chain**

* Initially FPGA hardware modules process the data and perform multiplexing (take signals from various sources and consolidate into one signal) and time-slice building
* The data are then processed by readout cards, accompanied by readout software
* HLT performs the final data filtration
* Several parallel streams run at each level (the system is highly scalable)

**Data Structure**

* Custom data protocol is devised. This consists of layers of encapsulated data
* At the fundamental level, data are divided into **images (data from a single detector)** within a given time interval (which depends on the detector time response)
* Images are incorporated into **time slices (data from all detectors)** within a specific time interval

**HLT Framework**

* Distributed computational framework based on a master slave architecture
* Slices are processed on many nodes and threads in parallel
* Written in C++ and the Qt framework (for the GUI)
* Advanced libraries for inter-process communication, networking, databases, algorithms, etc.
* The algorithms are modular and are highly optimised (possibility to implement new filter algorithms)

**Filtering Steps**

1. Detector information is extracted for filter decision
2. Data that are used for filtration are decoded and analysed in the time dimension, following which the spatial properties are considered
3. A binary decision is then made, on whether to keep the data or to drop it
4. The decision is applied on raw data, thereby reducing its size
5. Data is written to an output file

**Continuous Time Calibration**

* HLT filter process and decoding of trigger hits must be aligned in time (to ensure continuity of processing)
* HLT is able to compensate for time drifts within a single image width
* If hits are shifted by more than a single image width, recalibration is required
* The newly computed coefficients are then loaded back to the frontend cards

**Identification of Event Candidates**

**Data Reduction**

* If an event candidate is identified in the data, two images of the detector are saved
* The last image of a slice is also copied to the next slice to prevent edge cases (extrema)
* All images that are not associated with any event candidates are removed
* Eventually, HLT appends a list of valid events at the end of slice

**HLT Benchmarks**

* HLT filter was tested with artificial data using two servers and simple filtration algorithms
* Rate of processing depends on:
* **Number of threads**
* **Used filter algorithm**
* **Number of projections (planes)**
* **Slice Duration, etc.**