Status of the ProtoDUNE BSM physics trigger

ProtoDUNE-BSM meeting

April 11th, 2024

Hamza Amar Es-sghir

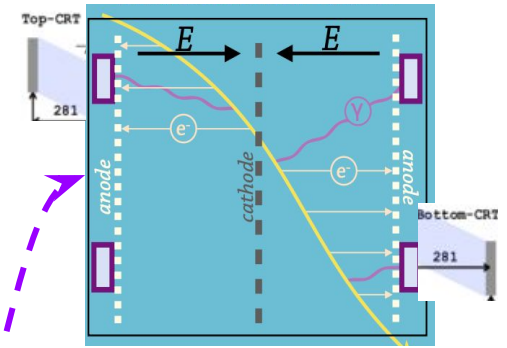
Animesh Chatterjee

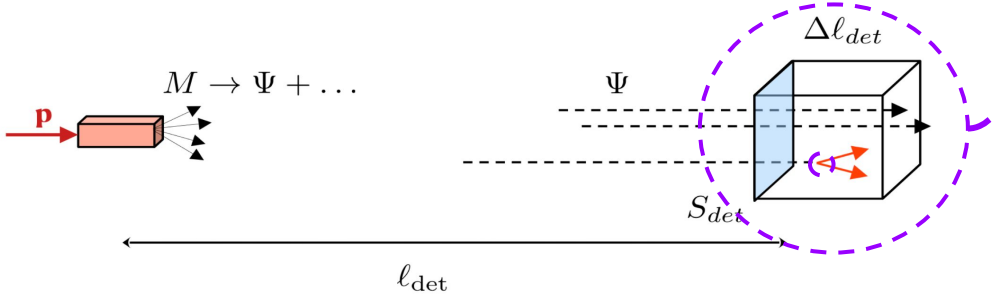
DEEP UNDERGROUND NEUTRINO EXPERIMENT

VNIVERSITAT DE VALÈNCIA

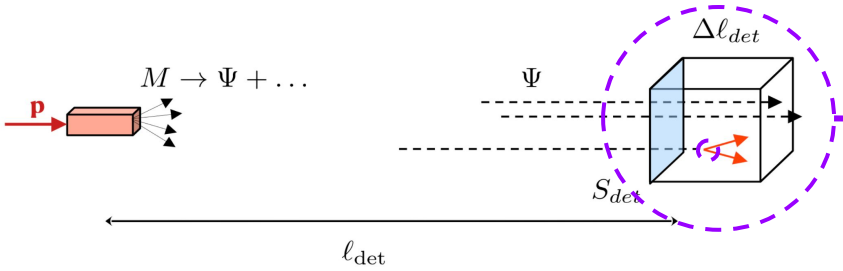
Motivation of the BSM trigger

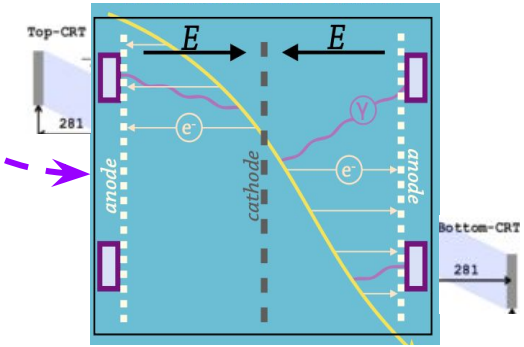
- Since ProtoDUNE detector is on the surface, cosmic ray (CR) muons are the dominant background for very low-rate signatures.
- Need to develop new trigger algorithms to minimize background and only search for events coming from the beam direction in time coincidence with the beam spill.
- Information from all readout planes is indispensable for directionality purposes. This allows for vertex reconstruction of events.
- Background suppression based on long track selections as well as directionality cuts.
- Available data to test algorithms has been VD-Coldbox data from several runs.





Key information for trigger purposes





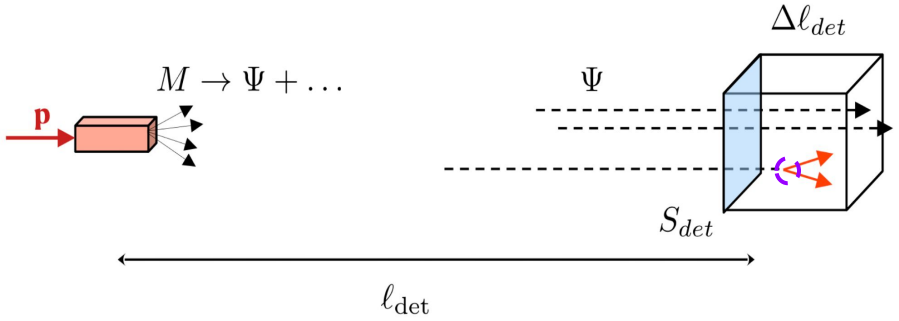
Need to save the beam timestamp to be synchronized with the protons hitting T2: spill starts and spill ends.

Relevant information for triggering

- **TPC signal**: hit information (channel wire, charge and time).
- **CRT signal**.
- At the moment there is **no light information**.

Key components of the BSM trigger

- BSM Trigger will be developed based on three key components
  - **Rejection of cosmic rays:** as a proof-of-principle we want an algorithm to identify long cosmic muon tracks (mostly vertical).
  - **Directionality:** selecting events which are coming from mostly beam direction.
  - **Time coincidence with beam spill duration**: trigger algorithms intended to minimize background by only searching for events in time coincidence with the beam spill (4.8 s).



Cosmic ray rejection: adjacency trigger algorithm

Adjacency stands for consecutive TP channels for a given track

Trigger Primitives (TPs)

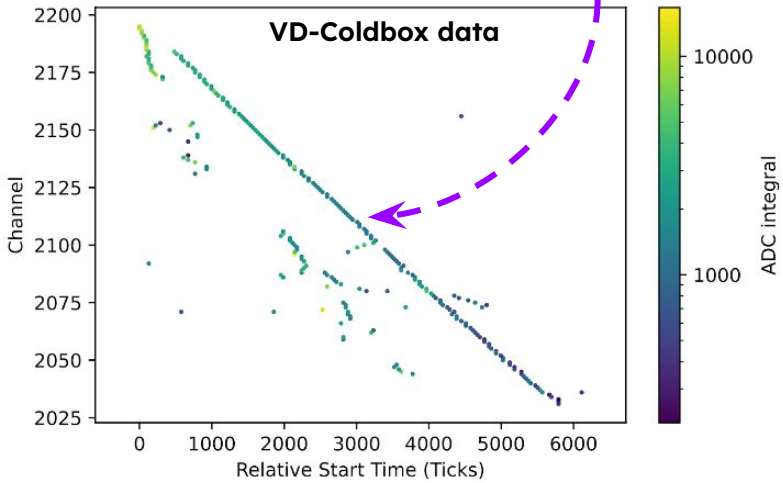
Adjacency trigger (based on adjacent hit collection wires threshold)

Goal: find longest track within a time window of 8000 ticks. Time and channel information are combined

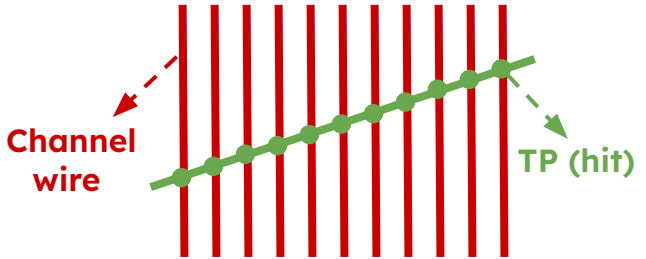
Channel adjacency TPs at adjacent channels. Channel tolerance accounts for TP skips.

Time tolerance TPs must also meet a time tolerance to be considered adjacent, i.e. of the same track.

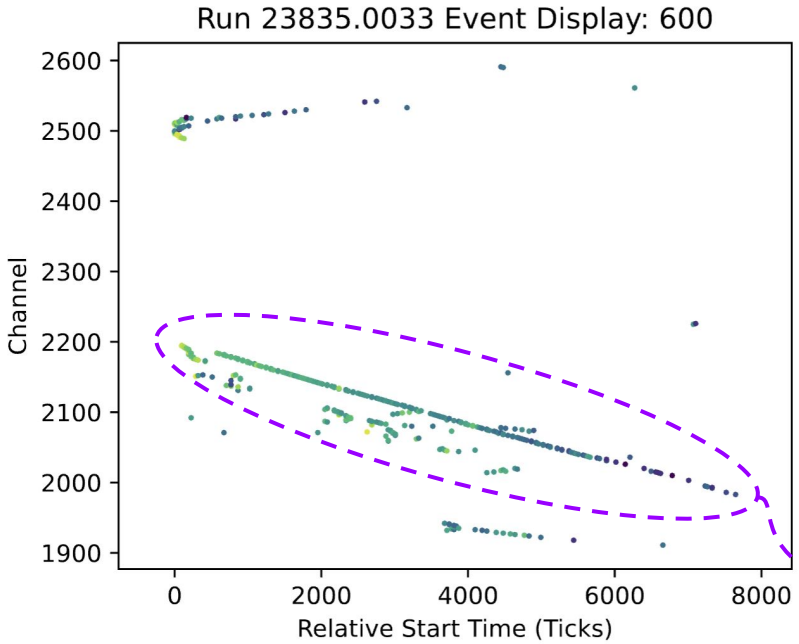
Trigger Activity (TA)



Track projection on readout plane wires

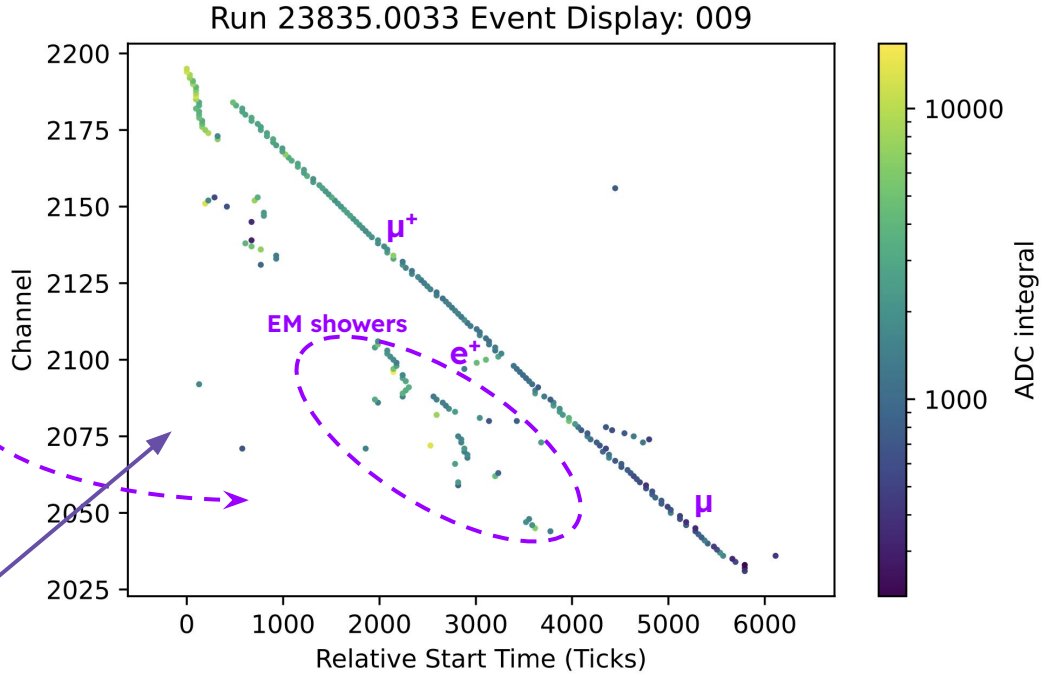


Cosmic ray rejection: new trigger algorithm performance

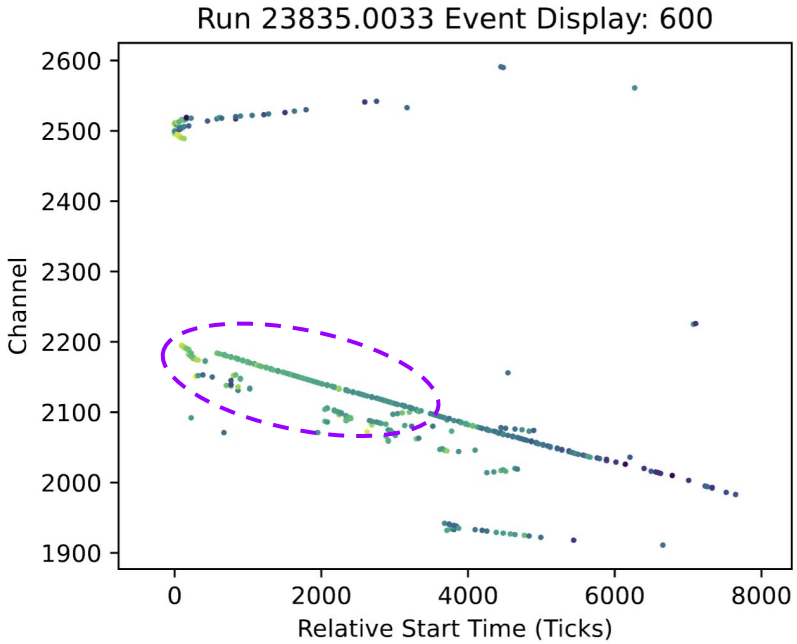


Inputs	**Adjacency threshold**: 100 channels **Time tolerance**: time window length **Adjacency tolerance**: 0 channels

Before changes



Cosmic ray rejection: new trigger algorithm performance

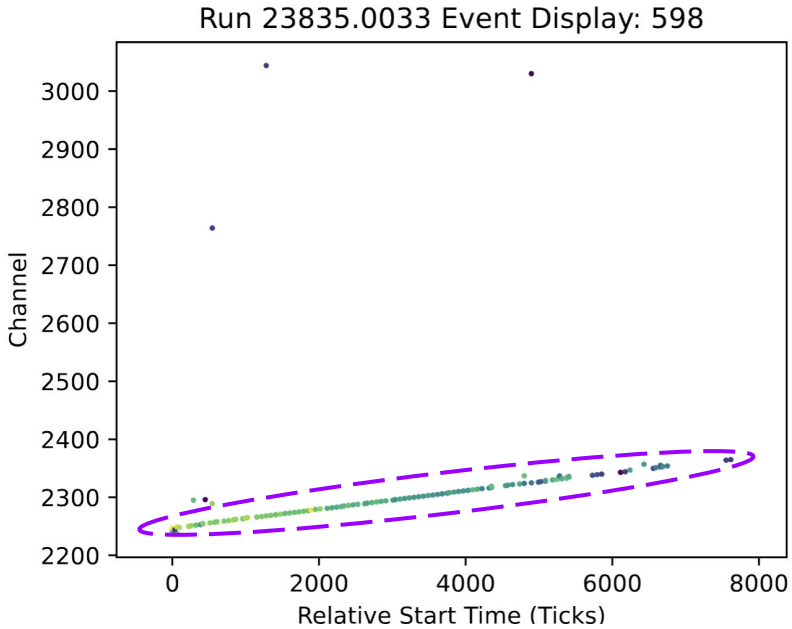


Inputs

Adjacency threshold: 80 channels
Time tolerance: 120 ticks
Adjacency tolerance: 5 channels

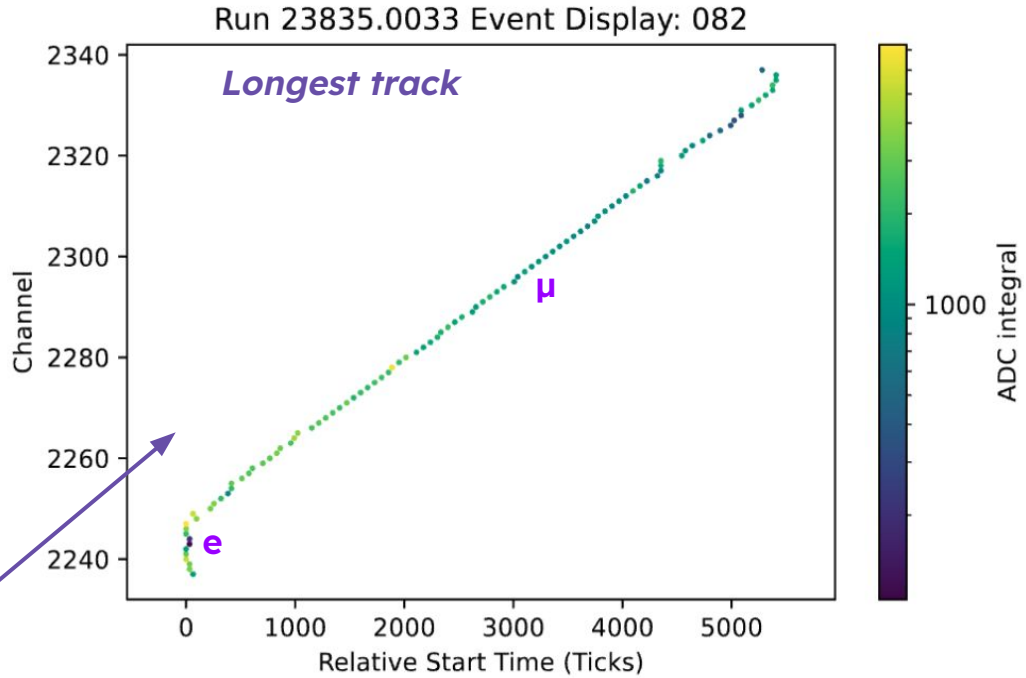
After changes

Cosmic ray rejection: new trigger algorithm performance



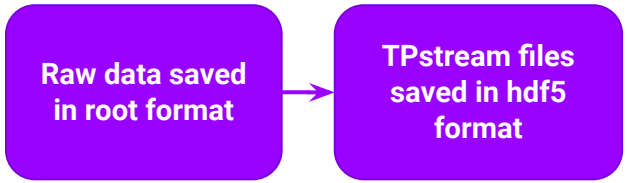
Inputs	**Adjacency threshold**: 100 channels **Time tolerance**: 100 ticks **Adjacency tolerance**: 3 channels

After changes



Takeaways for the adjacency algorithm

- Combination of both TP channel and time information has proven an efficient option to select muons in the VD-Coldbox.
- Instead of saving the full time window, solely the longest muon track is saved.
- A posterior combination with a directionality trigger will reduce the activity of the detector searching for events along the beam direction to complete the background suppression.
- Relevant parameters to be considered are:
  - Adjacency threshold.
  - Time tolerance.
  - Adjacency tolerance.



- Next tests and the estimation of the CR suppression will be carried out with previous ProtoDUNE-SP data after proper conversion of raw data into Trigger Primitives to test the algorithms.
- In case we wanted to save further tracks, not necessarily muons, in the same time window, code may be adjusted to fulfill the requirements.

Flowchart diagram for directionality algorithm

After adjacency algorithm to reduce the CR rate considering events coming from the beam direction

Trigger Activity (TA)

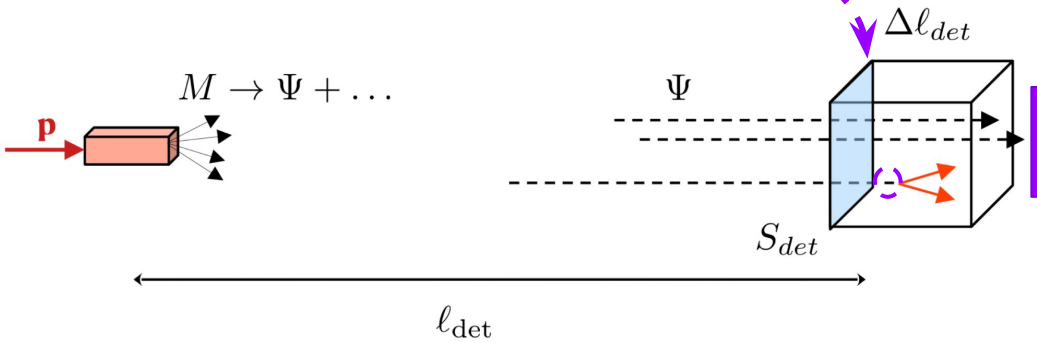
Directionality (tracks with a specific direction)

Orientation (φ) (with channel map position at the three readout planes)

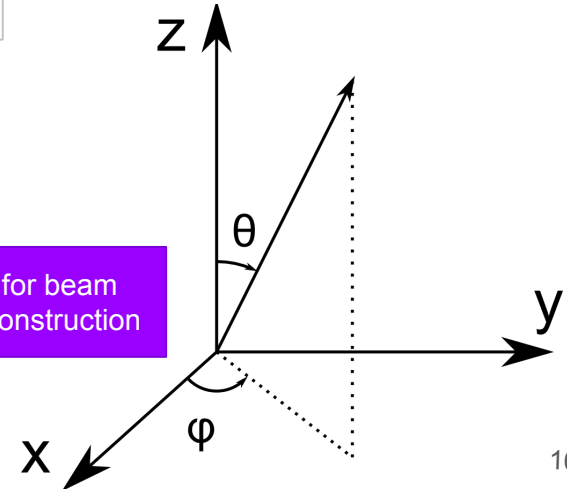
Verticality (θ) (from drift time at the collection plane and orientation)

What is the CR rate after rejection?

Trigger Candidate (TC)



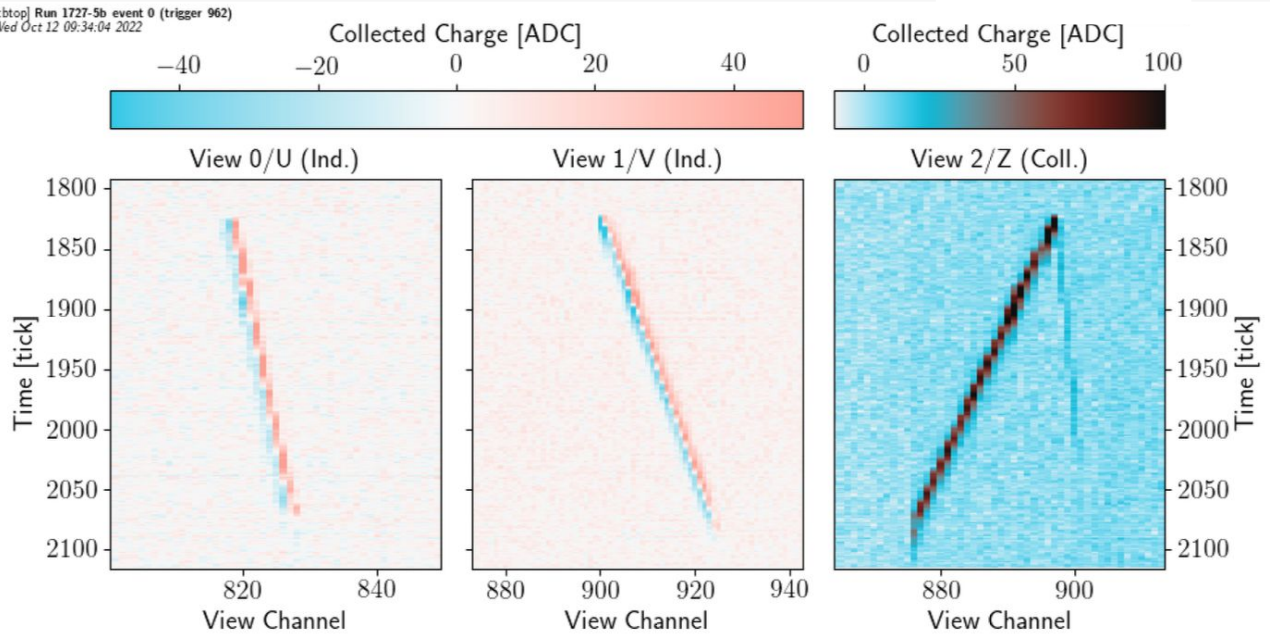
Indispensable for beam event vertex reconstruction



Directionality trigger algorithm

A single readout plane is not enough to infer incoming track direction.

- Verticality will depend on the orientation of the track projection on anode plane (i.e. wrt readout wires).
- Relative drift time is directly related to drift length, yet not channel.
- Merging information from all readout planes is essential to work out the track orientation and vertex reconstruction.
- Available tools to access TPs collected by each plane.



Trigger development timeline: current progress

Introduction to trigger tools

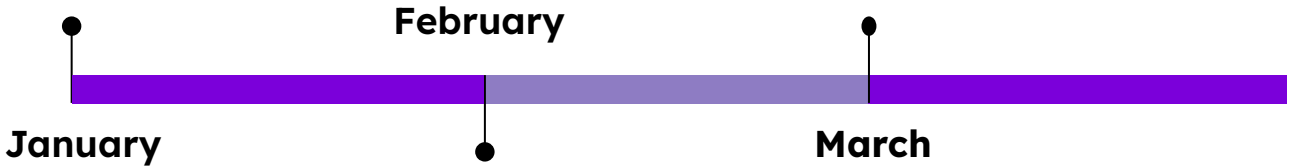
Introduction to trigger algorithms and their structure.

Discussion with TDAQ experts at CERN for orientation.

Channel-time adjacency algorithm

Development of an enhanced adjacency algorithm to select long tracks.

Specially intended for cosmic ray muon suppression.



Presentation of the plans for BSM trigger at the DAQ general meeting

Approval of a special trigger for protoDUNE-BSM searches at the DUNE-DAQ meeting.

Collaboration with the Technical Trigger Working Group to improve algorithms related to muon selection.

Trigger development timeline: future prospects

Directionality trigger

Develop trigger activity makers for both induction planes based on adjacency.

Infer track verticality and orientation combining hit information from all readout planes.

ProtoDUNE-HD run

Test algorithms with online beam events.

Event reconstruction with LArSoft to validate data.

If a high CR rate is seen, a workaround is needed. CRT (anti)coincidences is an option to be considered for an external triggering on CR muon tracks.

Reduce the cosmic ray rate in ProtoDUNE

Test algorithm performance selecting reconstructed CR events by using PD-SP data.

Check how many background events remain when crossing parallel to the cathode plane (beam direction).

Demonstrate that the cosmic rate can be attenuated.

Open questions for the TDAQ experts

- How to convert root data into hdf5 extension to test trigger algorithms with ProtoDUNE-SP data?

- For the directionality trigger we will need TPs of both induction and the collection planes. There are two options:
  - Create a single TA Maker for all three planes.
  - A single TA Maker for each plane and combine them in a single TC.

  In discussion with Trigger WG.

- In case the CR rate was high, combine with CRT information to see if is possible to further reduce it.

- When the Module Trigger Level makes a trigger decision passed on the Readout all data (TPs, TAs and TCs) will be stored to be used for offline analysis, right? In this way we would not need to flag BSM data.

- Exploit eventually PDS information if available for trigger purposes.

Summary and conclusion

- We have a clear idea on how to develop trigger for BSM signatures at ProtoDUNE.
- Algorithm to reject cosmic rays are in place and tested with VD-Coldbox data:
  - Need to validate it with protoDUNE-Run1(SP) data.
- Acquire data with BSM trigger during the next protoDUNE-NP04 run.
- Started working to develop trigger algorithm tools to include directionality.
- Include beam timing info within the trigger.
- Measure the trigger efficiency to reject cosmic ray after including all the three component.

Backup

New algorithm logic

```
uint16_t
TriggerActivityMakerAdjacency::check_adjacency()
{
  uint16_t adj = 1;              // Initialize adjacency, 1 for the first wire
  uint16_t max = 0;              // Maximum adjacency of window, which this function returns
  unsigned int channel_diff;     // Channel difference between to consecutive TPs of a ChannelID ordered TP vector
  unsigned int index;            // Index of the previous TP satisfying the adjacency condition
  int64_t start_time_diff;       // Start time difference between to consecutive TPs of a ChannelID ordered TP vector
  unsigned int tol_count = 0;    // Tolerance count, should not pass adj_tolerance

  std::vector<TriggerPrimitive> tp_inputs = m_current_window.inputs;
  std::vector<TriggerPrimitive> tp_track;
  std::vector<TriggerPrimitive> tp_longest_track;

  // Generate a channelID ordered list of hit channels for this window: m_current_window
  std::sort(tp_inputs.begin(), tp_inputs.end(), [](const TriggerPrimitive& tp1, const TriggerPrimitive& tp2) {
    return tp1.channel < tp2.channel;
  });
```

New algorithm logic

```
// Loop over the TPs of the current window
for (unsigned int i=0; i < tp_inputs.size(); i++){
  // Initiate/Resume the search with the first/a TP of the current window
  tp_track.push_back(tp_inputs.at(i));
  // index corresponds to the position of the latest TP of the current track
  index = i;
  unsigned int j=i+1; // j index to compare TPs further than the TP at position index
  channel_diff = 0; // To access the while loop for every iteration of i
  // Loop over the TPs of the current window to find the longest track
  while (j < tp_inputs.size() && (channel_diff == 1 || channel_diff == 0)){
    channel_diff = tp_inputs.at(j).channel - tp_inputs.at(index).channel;
    start_time_diff = std::abs(static_cast<int64_t>(tp_inputs.at(j).time_start) - static_cast<int64_t>(tp_inputs.at(index).time_start));
    // Check adjacency from that TP at position i. Add next TP and adjacency if the condition is fulfilled
    if (channel_diff == 1 && start_time_diff < m_time_tolerance){
      tp_track.push_back(tp_inputs.at(j));
      adj++;
      index = j;
    }
    // Check adjacency from TP at position i. Add only next TP if the condition is fulfilled
    else if (channel_diff == 0 && start_time_diff < m_time_tolerance){
      tp_track.push_back(tp_inputs.at(j));
      index = j;
    }

    // If next channel is not on the next hit, but the 'second next', increase adjacency
    // but also tally up with the tolerance counter.
    else if ((channel_diff == 2 && start_time_diff < 2*m_time_tolerance) || (channel_diff == 3 && start_time_diff < 3*m_time_tolerance)
    || (channel_diff == 4 && start_time_diff < 4*m_time_tolerance) || (channel_diff == 5 && start_time_diff < 5*m_time_tolerance)
             && (tol_count < m_adj_tolerance)) {
      tp_track.push_back(tp_inputs.at(j));
      adj++;
      index = j;

      for (unsigned int i = 0 ; i < channel_diff ; i++) tol_count++;
      channel_diff = 0; // To stay in the while loop
    }

    j++;
  } // End of while loop
  // Verify if new adjacency exceeds the max value. If so, max and tp_longest_track are updated.
  if(adj > max){
    max = adj;
    tp_longest_track = tp_track;
  }
  // Update the counter and reset tp_track (TP vector)
  adj = 1;
  tp_track.clear();
} // End of for loop

// Add the TPs that caused the adjacency to be the global maximum after channel-time filtering
for (auto tp : tp_longest_track) {
  m_longest_track_window.add(tp);
}

return max;
}
```

Variable	Role	Name
Channel adjacency	Accounts for channel multiplicity	*adj*
Time tolerance	Max time difference admitted for adjacent channels	*m_time_tolerance*
Adjacency tolerance	Permitted channel skips	*m_adj_tolerance*

Trigger Primitive structure

4.6.3.1 Trigger Primitive Generation

The implementation of Trigger Primitive generation algorithms, although a Data Selection system subcomponent, falls under the responsibility of the Readout System. The algorithm design and output format specification, however, falls under the Data Selection system. The input to the TPC and PDS primitive generation algorithms is raw unbiased TPC data, and minimally biased PDS data, respectively. The output of the TPC and PDS is Trigger Primitives, which for TPC data is shown here:

```
struct TriggerPrimitive
{
  version_t version = s_trigger_primitive_version;
  timestamp_t time_start = INVALID_TIMESTAMP;
  timestamp_t time_peak = INVALID_TIMESTAMP;
  timestamp_t time_over_threshold = INVALID_TIMESTAMP;
  channel_t channel = INVALID_CHANNEL;
  uint32_t adc_integral = { 0 };
  uint16_t adc_peak = { 0 };
  detid_t detid = INVALID_DETID;
  Type type = Type::kUnknown;
  Algorithm algorithm = Algorithm::kUnknown;
  Flags flag = 0;
};
```

To provide good sensitivity to different track topologies, each Trigger Primitive contains information such as the time-over-threshold of the waveform, its peak, its total charge, as well as the timestamp of the start of the waveform.

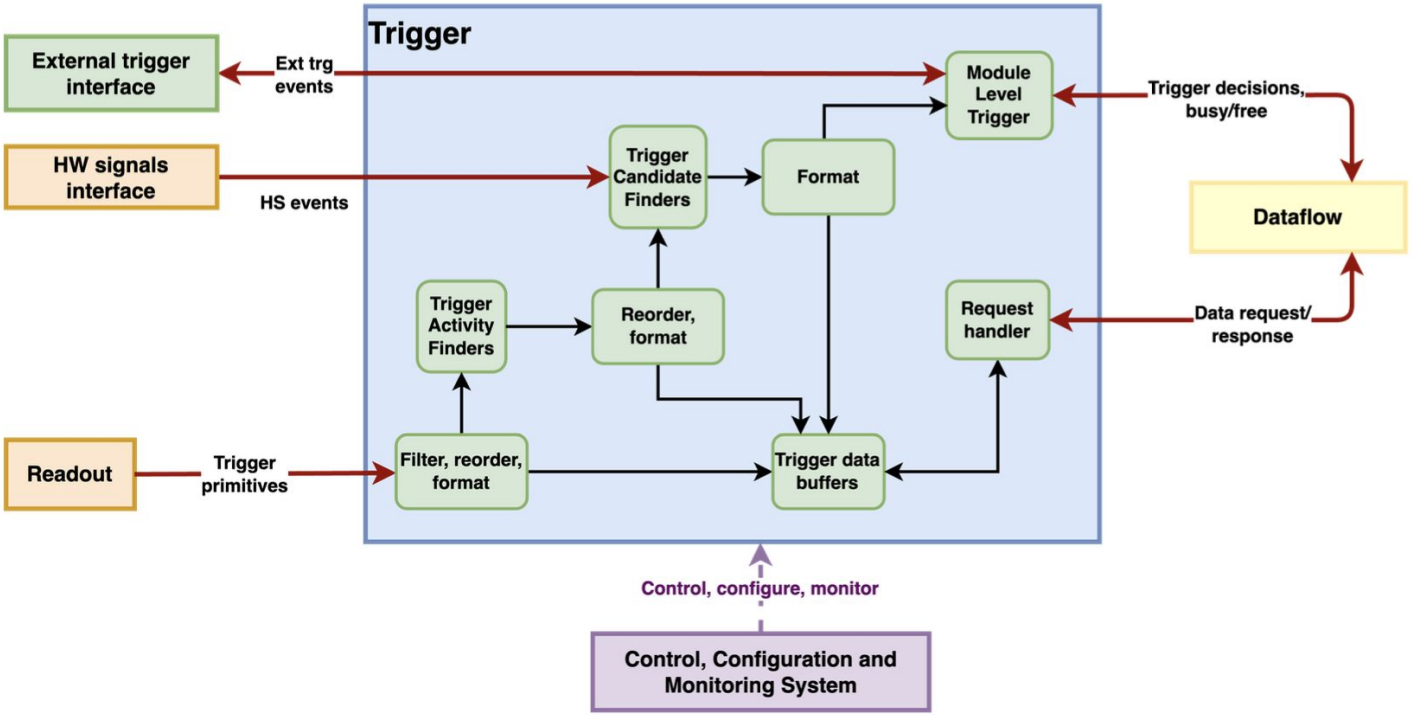
Here, `type` can be `kTPC` or `kPDS`, and `algorithm` represents the particular algorithm used to find the Trigger Primitive. Note that while it is possible to use the Trigger Primitives in this form for storing with a Trigger

Trigger Activity structure

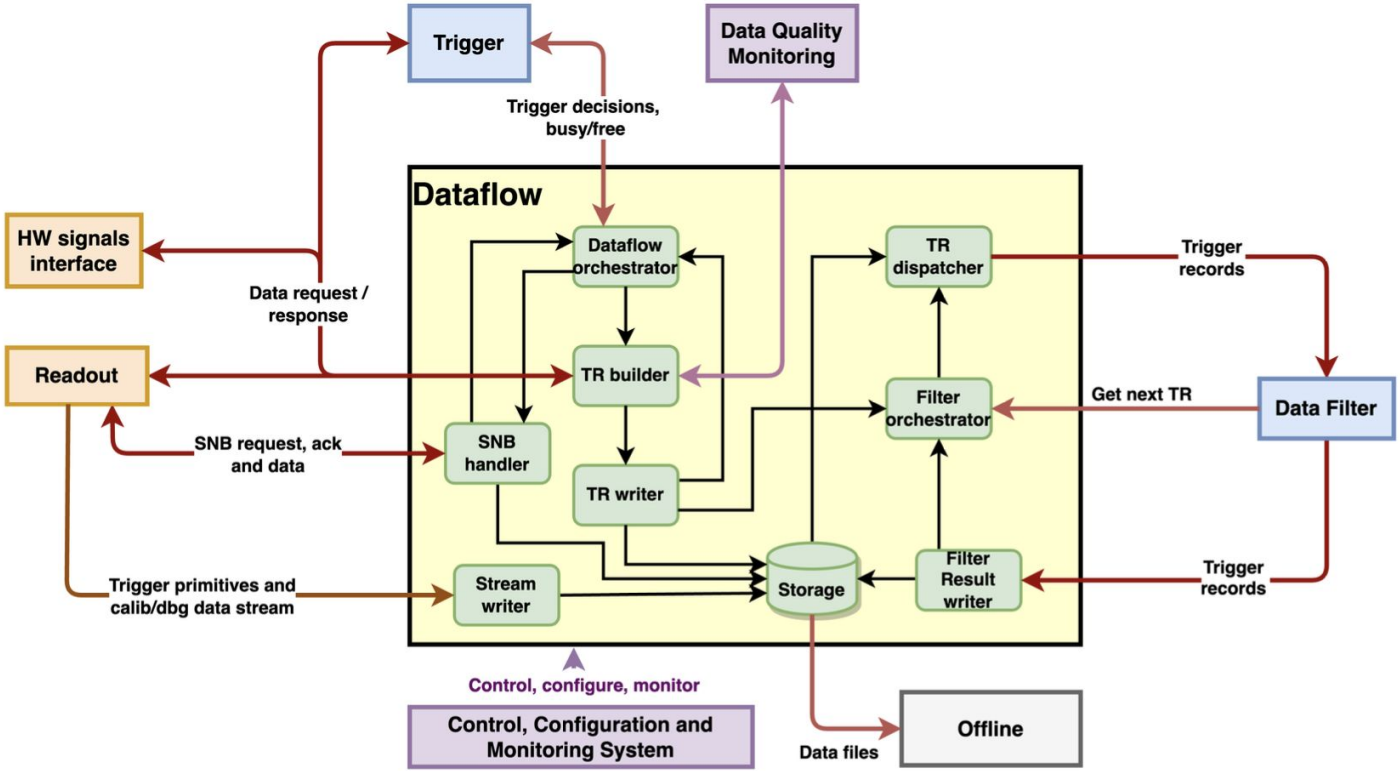
```
struct TriggerActivity
{
  static constexpr version_t s_trigger_activity_version = 1;

  version_t version = s_trigger_activity_version;
  timestamp_t time_start = INVALID_TIMESTAMP;
  timestamp_t time_end = INVALID_TIMESTAMP;
  timestamp_t time_peak = INVALID_TIMESTAMP;
  timestamp_t time_activity = INVALID_TIMESTAMP;
  channel_t channel_start = INVALID_CHANNEL;
  channel_t channel_end = INVALID_CHANNEL;
  channel_t channel_peak = INVALID_CHANNEL;
  uint64_t adc_integral = 0;
  uint16_t adc_peak = 0;
  detid_t detid = INVALID_DETID;
  Type type = Type::kUnknown;
  Algorithm algorithm = Algorithm::kUnknown;
};
```

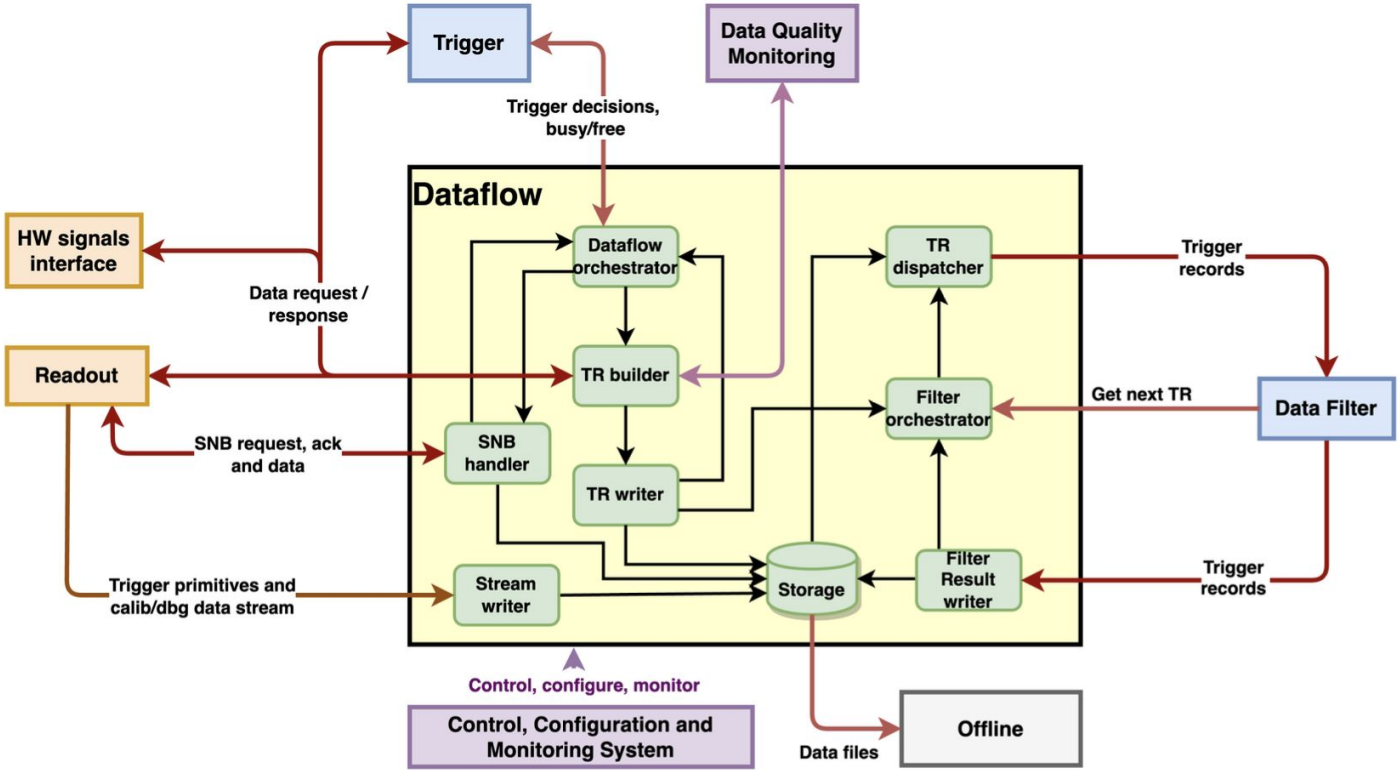
Trigger components and their interaction with other DAQ sub-systems



Interfaces of Dataflow components with other TDAQ sub-systems



Interfaces of Dataflow components with other TDAQ sub-systems



Readout planes for Horizontal drift module

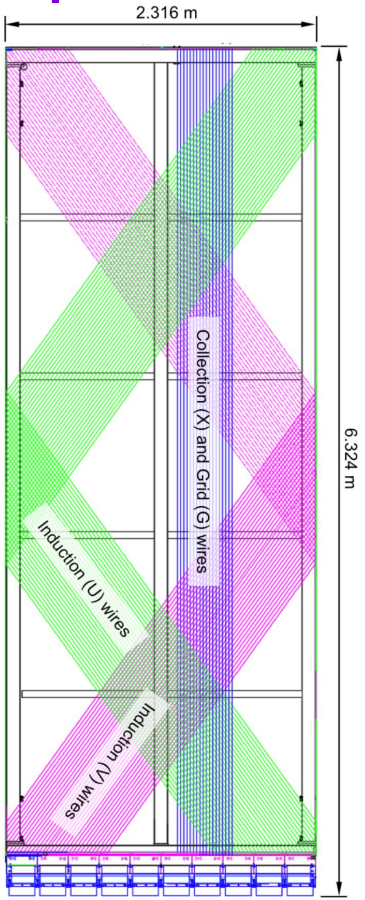


Table 2.3: APA design parameters

Parameter	Value
Active height	5.984 m
Active width	2.300 m
Wire pitch (U, V)	4.7 mm
Wire pitch (X, G)	4.8 mm
Wire pitch tolerance	$\pm$0.5 mm
Wire plane spacing	4.8 mm
Wire plane spacing tolerance	$\pm$0.5 mm
Wire Angle (w.r.t. vertical) (U, V)	$\pm$35.7°
Wire Angle (w.r.t. vertical) (X, G)	0°
Number of wires / APA	960 (X), 960 (G), 800 (U), 800 (V)
Number of electronic channels / APA	2560
Wire material	beryllium copper (CuBe)
Wire diameter	152 μm

Readout planes for Vertical drift module