A cluster-finding algorithm for free-streaming data

Volker Friese for the CBM Collaboration

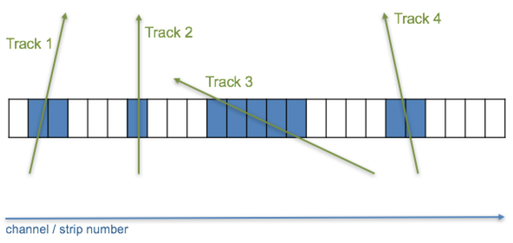
GSI Helmholtzzentrum für Schwerionenforschung, Darmstadt, Germany

CBM

Compressed Baryonic Matter experiment at FAIR

A cluster finder - so what?

- Consider a position-sensitive device - say, for instance, a silicon strip sensor.
- Within one event (collision), it is traversed by a number of particles (tracks). Depending on its inclination, each track activates one or more adjacent strips - a cluster.
- The first step of data reconstruction is to find these clusters from the raw data and determine the cluster position, which is the measurement of the intersection point of the particle trajectory with the detector.



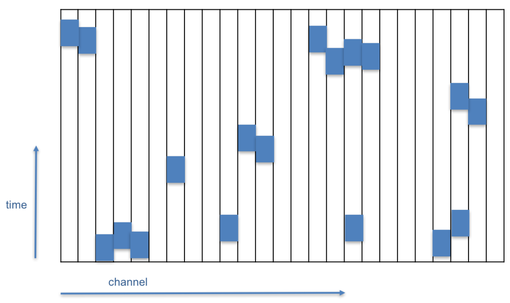
So what? you say. Simple problem, isn't it? Just a loop over the channels, 0-1 transitions mark the start of the cluster, 1-0 transitions its end. A no-brainer, a ten-liner. Has been done a million times.

Right.

Free-streaming data - what is that?

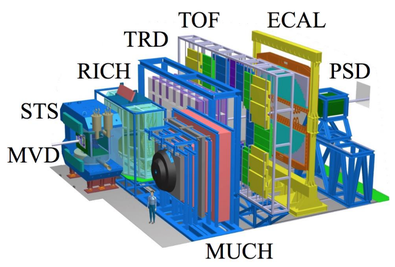
- Now imagine there is no external trigger to your detector electronics. It triggers itself, meaning whenever a channel is activated (above a given charge threshold), it sends a message to the data acquisition. Let's call the message a "digit". It contains just the channel address, the digitized charge, and a time stamp.
- What you get as input for reconstruction is then a stream of digits without association to events (collisions).
- Let's add an extra complication: events may not well be separated in time. In other words, they come with a frequency comparable to the inverse time resolution of the detector.

Suddenly, the problem of cluster finding is not so trivial any more. It becomes two-dimensional (in address space and time).

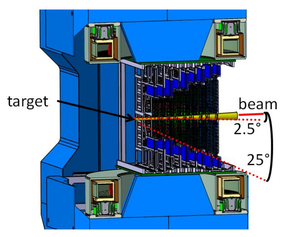


Who does such weird things?

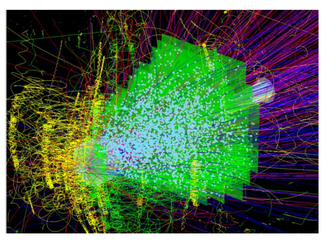
- CBM: a heavy-ion experiment at the new FAIR facility in Darmstadt. Under construction; operation starting 2024.
- Design goal: very high interaction rates (10^7 / s)
- Self-triggered read-out electronics, free-running DAQ.
  Event reconstruction and -selection in real-time.
- Silicon Tracking System: track reconstruction in a dipole field; about 900 double-sided silicon strip sensors.



CBM detector setup



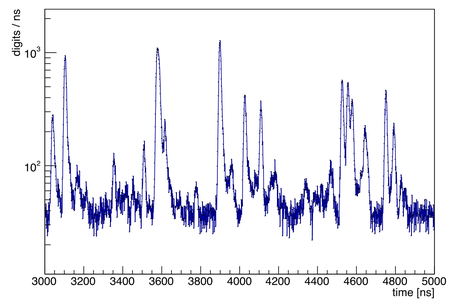
Silicon Tracking System



Typical event in the STS (Au+Au @ 10A GeV/c)

The shape of input data and the challenge

- The input is a time-stream of digits; their association to events is not a-priori known.
- Digits are packed into transport containers called "time-slices", typically containing several thousand events.
- Events come randomly; they may overlap in time.
- Cluster finding must be very fast, such that it can be performed in real-time!



Number frequency of STS digits in a small fraction of a time-slice at 10 MHz interaction rate (Au+Au at p = 10A GeV/c).

First approach:

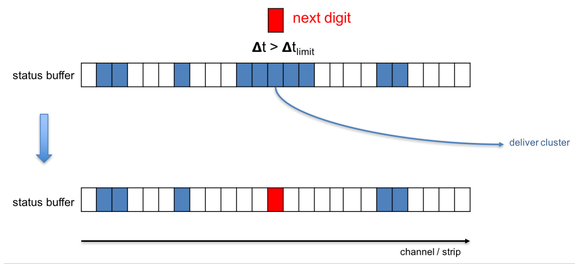
Discretize the time axis and do 2-d cluster finding.

That works for sure - but it is not performant.

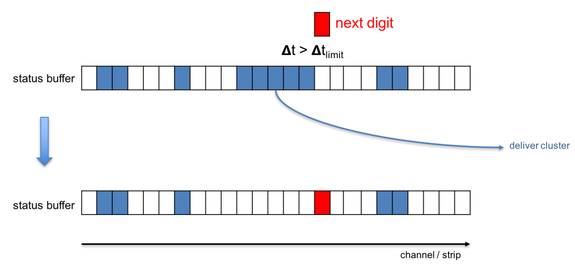
Let's try to find a "free-streaming" algorithm which matches the "free-streaming" input data.

The algorithm

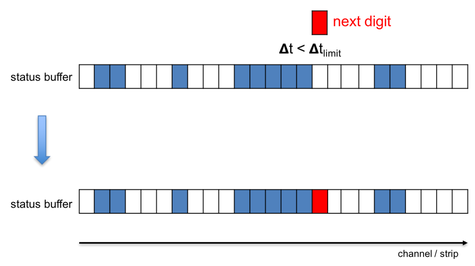
- Two digits are considered to belong to a cluster if they are in neighbouring channels and their time difference is smaller than the threshold $3\sqrt{2}\,\sigma_t$.
- A running status of the sensor is monitored: time of last digit in each channel and a link to the digit object. The status buffer is realised as vector<double, int>.
- The digits from the input stream are processed one after the other. It is assumed that they are delivered time-ordered.
- After all digits are processed, the remaining clusters in the buffer are delivered.



Case 1: The buffer already has a digit in this channel. The time difference to the new digit is larger than the threshold. Deliver the cluster, delete the corresponding digits from the buffer, and add the new one.



Case 2: The buffer has a digit in a neighbouring channel. The time difference to the new digit is larger then the threshold. Deliver the cluster, delete the corresponding digits from the buffer, and add the new one.

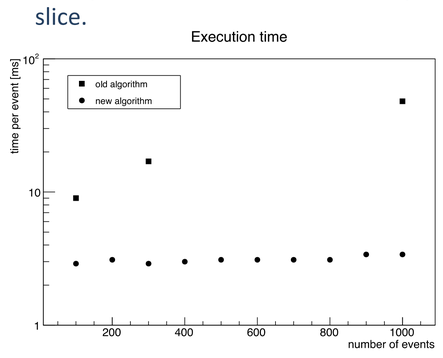


Case 3: The buffer has a digit in a neighbouring channel. The time difference to the new digit is smaller then the threshold. Add the new digit to the buffer.

Simple again, isn't it? Not a ten-liner, but a fifty-liner. Yet very effective.

Features and Performance

- The algorithm requires time-sorted input. This will be achieved either in the DAQ on FPGA level or in prior sorting in software.
- The output clusters are not time-sorted, since the moment when a cluster can be delivered depends on the (random) arrival of the next digit in or next to the active channels.
- As expected, the execution time depends linearly on the number data (events) in a time slice.



- Execution time : 3 ms / event (min. bias Au+Au @ p = 10A GeV/c , on average 5,500 digits and 1,700 clusters per event).
- Independent of the size of the time-slice (number of input data).
- 55% of the time is spent on cluster finding, 45% on the cluster analysis (determination of cluster centre position and its error).
- For comparison, the speed of the old, "2d" cluster finder is shown.
- Measurements were performed on a single core of an Intel Core-i7 (about 22 HepSpec06).

Summary

- An algorithm for the reconstruction of clusters from a stream of input data not associated to events was developed.
- The algorithm works in the spirit of time-streaming data and avoids any loops and associated data containers.
- Its execution time is about 3 ms per Au+Au event on a single core and scales linearly with the number of input data (size of the time-slice). About half of the execution time is spend in cluster finding, the other half in the cluster analysis.
- Significant speed-up was achieved in comparison to a prior implementation performing a 2-dimensional cluster search in address space and time.
- Parallelisation on a many-core architecture is straightforward since each of the 900 sensors can be processed independently.

FAIR HELMHOLTZ RESEARCH FOR GRAND CHALLENGES GSI