A cluster-finding algorithm for free-streaming data

Volker Friese[1,*]

[1]GSI Darmstadt, Planckstraße 1, 64291 Darmstadt, Germany

Abstract. In position-sensitive detectors with segmented readout (pixels or strips), charged particles activate in general several adjacent read-out channels. The first step in the reconstruction of the hit position is thus to identify clusters of active channels associated to one particle crossing the detector. In conventionally triggered systems, where the association of raw data to events is given by a hardware trigger, this is an easy-to-solve problem. It, however, becomes more involved in untriggered, free-streaming read-out systems like the one employed by the CBM experiment. Here, the time coordinate of the single-channel measurement must be taken into account to decider whether neighbouring active channels belong to a cluster. A simple extension of well-known cluster finding algorithms is not satisfactory because of involving increasing combinatorics, which are prohibitive for reconstruction in real-time. In this article, a cluster-finding solution for the Silicon Tracking System of the CBM experiment is presented which avoids any combinatorics or loops over detector channels. Its execution time is thus independent on the size of the data packages (time slices) delivered by the data acquisition, making it suitable for being used in online reconstruction.

1 Introduction

Position-sensitive detectors reconstruct the intersection point of a particle from the charge distribution generated by the particle during its passage through the active detector material. The charge is collected at a readout surface segmented in one dimension (strips) or two dimensions (pixel, pads), the granularity of the segmentation being driven by considerations like resolution in coordinate space and occupancy. In general, several readout segments (channels) are activated by a single particle, allowing to determine the coordinate by an analysis of the charge measurements in those channels.

The first step in the reconstruction of the hit coordinate is thus the identification of a cluster of active channels associated to one particle crossing the detector. In a conventionally triggered readout system, this is a rather trivial problem, since the trigger defines a set of measurements corresponding to a single (or a limited number of) event, in which clusters can be searched for. Within a trigger, all measurements are considered simultaneous. The data set in which to search for neighbouring active channels is thus well defined. The task can usually be achieved by a single loop over all detector channels. The complexity of the problem is thus of the order of n_{channels}.

In untriggered, free-streaming readout schemes, the situation becomes more involved. The events are not sorted by triggers, but are continuously recorded. Thus, each measurement

*e-mail: v.friese@gsi.de

is characterised by its address (channel number) and by the acquisition time. Obviously, the complexity of the problem increases, which is of particular importance if the cluster finding algorithm should be suitable for application in real-time as part of online reconstruction. An example is the CBM experiment and its Silicon Tracking System (STS), consisting of double-sided silicon strip sensors. In this article, the cluster-finding algorithm developed for that system is described.

2 The CBM experiment

CBM [1] is a fixed-target heavy-ion experiment currently under construction and will take data at the FAIR accelerator facility [2] in Darmstadt, Germany. It comprises a number of detector systems for measuring nuclear collisions in the beam momentum range 3.5A – 12A GeV. The Silicon Tracking System (STS) of CBM [3] will reconstruct particle trajectories inside a dipole magnetic field. It is made of double-sided silicon strip sensors.

A focus of the CBM experiment is the ability to measure at very high interaction rates (up to 10 MHz) in order to open access to extremely rare probes. Data reduction in real-time is a prerequisite for such interaction rates. The trigger topologies, however, are complex and not realisable in trigger logic. Thus, CBM will not exploit any hardware trigger, but have a free-streaming readout where all data are pushed to a computer farm, on which the trigger signatures are evaluated in software. This requires partial event reconstruction in real-time. The performance of the corresponding reconstruction algorithms is therefore a critical issue.

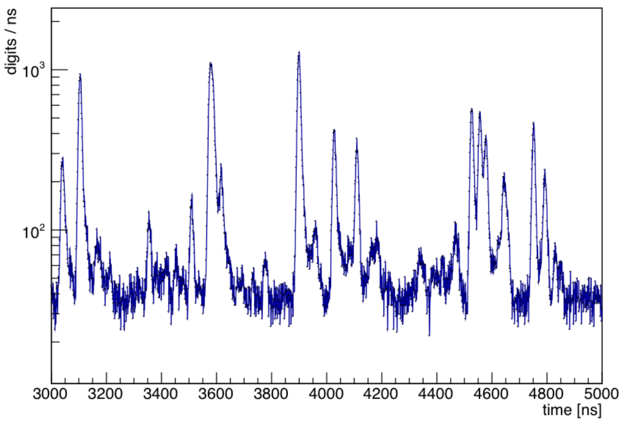

Figure 1. Simulated number frequency of raw measurements (”digis“) from the CBM-STS in s small fraction of a time-slice at 10 MHz interaction rate for Au+Au collisions at $p_{\mathrm{beam}} = 10A$ GeV.

The input data situation for the cluster finding problem is illustrated in Fig. 1, showing the frequency distribution of raw data (”digis“) from the STS system. Events can be seen as spikes in the distribution, but at high interaction rates, they may occur in close temporal proximity and are then not easily separable by the time measurements alone. For the transport through the data acquisition, data will be packed into ”time-slices“– containers typically comprising data from several thousands of collision events.

3 The algorithm

Figure 2 illustrates the algorithmic problem of cluster finding for free-streaming data from the STS. The problem is now of two-dimensional nature, the time measurement being the second coordinate besides the address (channel number). The additional condition for several measurements to belong to a cluster, besides being in adjacent channels, is that their time measurements coincide within a precision set by the detector resolution.

The simplest approach extends the one-dimensional approach by a double loop over all measurements, searching for each of them for a cluster partner. However, since the number of measurements in a time-slice is very large, this approach is prohibitive in terms of computational speed. The second obvious approach is to discretize the time axis and then perform a cluster-finding procedure on a two-dimensional grid. Because of the size of the input data (length of the time-slice, corresponding to several thousands of events), this would require the sub-division of the time-slice into smaller intervals, which may cause clusters to be artificially split between two sub-intervals.

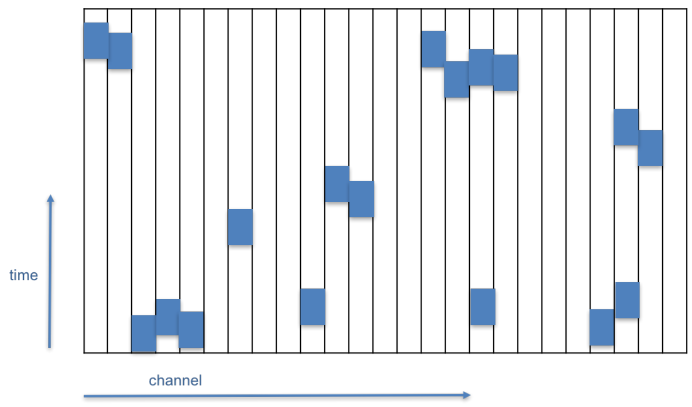

Figure 2. Illustration of the distribution of data in address and time space for a free-streaming readout

Taking into account the problems of these obvious approaches, a different cluster-finding procedure was chosen, which does not rely on pre-sorting of data into grids and subsequent cluster search, but on a continuous treatment of data. The basic considerations are:

- Two measurements are defined to belong to a cluster if they are in adjacent channels (strips) and their time difference is smaller than a threshold defined by the time resolution of the detector: $\Delta_{t,\mathrm{thr}} = 3\sqrt{2}\,\sigma_t$, with $\sigma_t = 5$ ns.
- Since the data are time-ordered, a cluster can be considered complete if there is a new measurement in one of its channels or one of its neighbouring channels not belonging to the cluster, i.e., not compatible in time.

The algorithm is based on bookkeeping of the state of each readout channel in the system. The number of channels is about 1.8 millions, dispersed over about 900 sensors. Each sensor can be treated separately. The bookkeeping keeps track of the last registered measurement in each channel. It is implemented as two `std::vector`, the sizes of which are predefined to equal the number of channels in order to avoid memory allocation after initialization. The first vector provides a pointer to the actual measurement, the second one keeps track of the measurement time. The vectors allow indexed access (O(0)) to the channel status. The memory expenditure is one integer and one double-precision float per channel, amounting to about 22 MB.

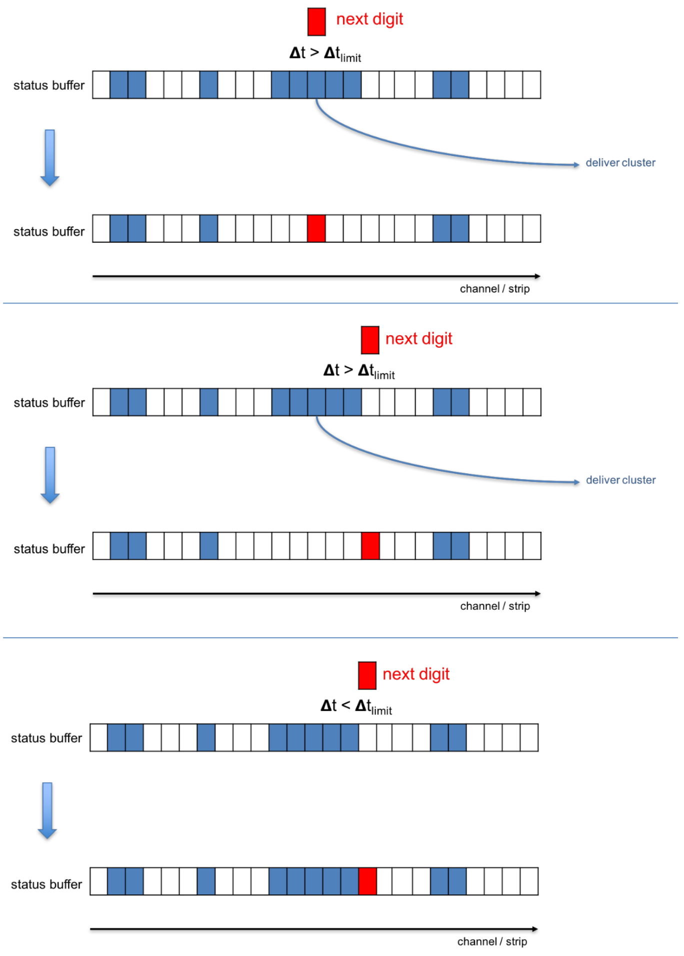

Figure 3. Working principle of the algorithm (see text). In each panel, the upper row shows the buffer status before, the lower row the status after the action taken on the incoming new measurement.

The measurements are treated one-by-one as delivered by the data acquisition. For each new measurement, the state of the respective detector channel and its two neighbours are looked at. The following cases can be distinguished (see Fig. 3):

1. The channel and its two neighbours are empty. The new measurement is added to the status.
2. The channel status shows a previous measurement (Fig. 3, top). A corresponding cluster object is created and delivered to further processing. All channels contributing to the cluster are cleared. The new measurement is added to the status vectors.
3. The channel status is empty, but one of the two neighbours (or both) are active. Depending on the time difference of the new measurements to that in the neighbour, there are two possibilities.

(a) The time difference is larger than the threshold (Fig. 3, centre). A cluster object is created from the neighbouring channel. The corresponding channels are cleared. The new measurement is added to the status.

(b) The time difference is smaller than the threshold (Fig. 3, bottom). The new measurement is added to the status.

Cluster creation from an active channel means that the left and right end of the cluster are determined by iteratively checking the status of the respective neighbours until an empty channel is found.

After the last measurement in the time slice is processed, there will be remaining data in the status buffer. From all these remaining data, clusters are created. After this final step, the status buffer is empty.

4 Features and performance

The working principle of the algorithm assumes that the data come time-ordered, which will be guaranteed by the data acquisition software during time-slice building. In contrast, clusters are not delivered time-ordered, since the time when a cluster is created depends on the (random) occurrence of a next measurement in a channel contributing to the cluster or in a neighbouring one. If later reconstruction steps (i.e., track finding) require time-sorted input, a sorting step will have to be introduced in between.

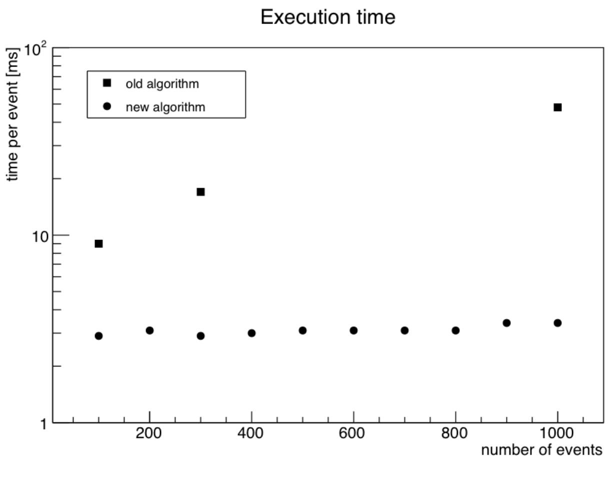

Figure 4. Algorithm execution time per event as function of time-slice length (number of events per time-slice), measured on single core of an Intel I7-4870HQ processor (2.5 GHz, 6 MB L3 cache). For comparison, the performance of the first, naive approach involving extensive loops over the detector data are also shown.

The performance of the algorithm was assessed by its application to simulated data from a full detector response simulation of typical events (minimum-bias Au + Au collisions at $p = 10A$ GeV/c) inside the CBM software system [4] using the FairRoot framework [5]. On average, one event contains about 5,500 measurements and 1,700 clusters. Since no loops over detector channels are involved, the execution time of the algorithm does not depend on the number of channels. From the construction of the algorithm – processing one measurement after the other – it can be expected that the execution time scales with the number of measurements. This expectation is met as demonstrated in Fig. 4, showing the execution time as function of the number of events in a time slice. About 3 ms per event are needed on a single core of an Intel I7-4870HQ processor (2.5 GHz, 6 MB L3 cache). This execution time comprises the cluster analysis (determination of cluster centre position and average time [6]), which again depends linearly with the number of clusters and is not connected to the algorithm described here. The mere cluster finding takes 45 % of the total time, i.e., about 1.5 ms.

5 Summary

An algorithm for cluster finding from free-streaming input data as delivered by the CBM silicon-strip detectors was presented. The algorithm does not require the association of measurements to events by an event trigger. As the input data, the algorithm is "free streaming": it processes the incoming data one-by-one. It avoids any loops over detector channels and associated data containers, such that the execution time is linear in the number of measurements. The algorithm is sufficiently fast for being applied in real-time data reconstruction. Its execution time is about 1.5 ms per minimum-bias event on a single core. Data-level parallelisation is straightforward since each of the about 900 sensors can in principle be processed independently.

The algorithm solves the one-dimensional problem as relevant for strip-like detectors. The extension to two-dimensional channel topographies (pixel / pad) should be straightforward by suitable definition of neighbouring channels. In that case, two-dimensional status bookkeeping will maintain the O(0) access to the single-channel status.

References

[1] T. Ablyazimov *et al.* (CBM collaboration), Eur. Phys. J. **A 53**, 60 (2017)
[2] `fair-center.eu`
[3] J. Heuser, Acta Phys. Polon. Suppl. **9**, 221 (2016)
[4] V. Friese (for the CBM collaboration), J. Phys. Conf. Ser. **898**, 112003 (2017)
[5] M. Al.Turany *et al.*, J. Phys. Conf. Ser. **396**, 022001 (2012)
[6] H. Malygina and V. Friese, J. Phys. Conf. Ser. **898**, 042022 (2017)