The high-rate data challenge: computing for the CBM experiment

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 112003 doi:10.1088/1742-6596/898/10/112003

The high-rate data challenge: computing for the CBM experiment

V Friese (for the CBM collaboration)

GSI Helmholtzentrum für Schwerionenforschung, Planckstraße 1, 64291 Darmstadt, Germany

E-mail: v.friese@gsi.de

Abstract. The Compressed Baryonic Matter experiment (CBM) is a next-generation heavy-ion experiment to be operated at the FAIR facility, currently under construction in Darmstadt, Germany. A key feature of CBM is very high interaction rate, exceeding those of contemporary nuclear collision experiments by several orders of magnitude. Such interaction rates forbid a conventional, hardware-triggered readout; instead, experiment data will be freely streaming from self-triggered front-end electronics. In order to reduce the huge raw data volume to a recordable rate, data will be selected exclusively on CPU, which necessitates partial event reconstruction in real-time. Consequently, the traditional segregation of online and offline software vanishes; an integrated on- and offline data processing concept is called for. In this paper, we will report on concepts and developments for computing for CBM as well as on the status of preparations for its first physics run.

1. Introduction: the CBM experiment

CBM is a heavy-ion experiment which will investigate strongly interacting matter at high net-baryon densities by studying collisions of heavy nuclei at moderate collision energies (2 – 45 GeV per nucleon) [1]. It will take external beams from the synchrotrons of the FAIR accelerator complex currently under construction in Darmstadt, Germany. The experiment is in the construction stage; first data taking is planned for 2024.

The experimental setup of CBM is depicted in figure 1. It comprises a number of sub-detectors, namely the Silicon Tracking System (STS) and the Micro-Vertex Detector (MVD) inside a dipole magnet for track reconstruction, RICH and TRD detectors for electron identification, a TOF detector for hadron identification, an electromagnetic calorimeter for the measurement of neutral probes, and a forward calorimeter (PSD) for event characterisation in terms of centrality and reaction plane angle. The RICH detector can be replaced by the muon system for measurements involving muons. A characteristic of the setup is versatility: it is foreseen to exchange or move components according to specific physics aims.

The experimental programme of CBM is manifold and comprises the measurement of a large number of observables [2]. Among them are extremely rare ones, e.g., multi-strange anti-hyperons, open and hidden charm. The multiplicity of such probes is expected to be one in a million collisions or even less (figure 2, left). In order to be sensitive to such rare observables, a key design feature of CBM is to be able to take very high interaction rates of up to 10^7 collisions per second. This is several orders of magnitude higher than the capacity of current or other planned experiments in the field of relativistic heavy-ion physics (figure 2, right) and poses extreme requirements for detectors, read-out electronics and data handling.

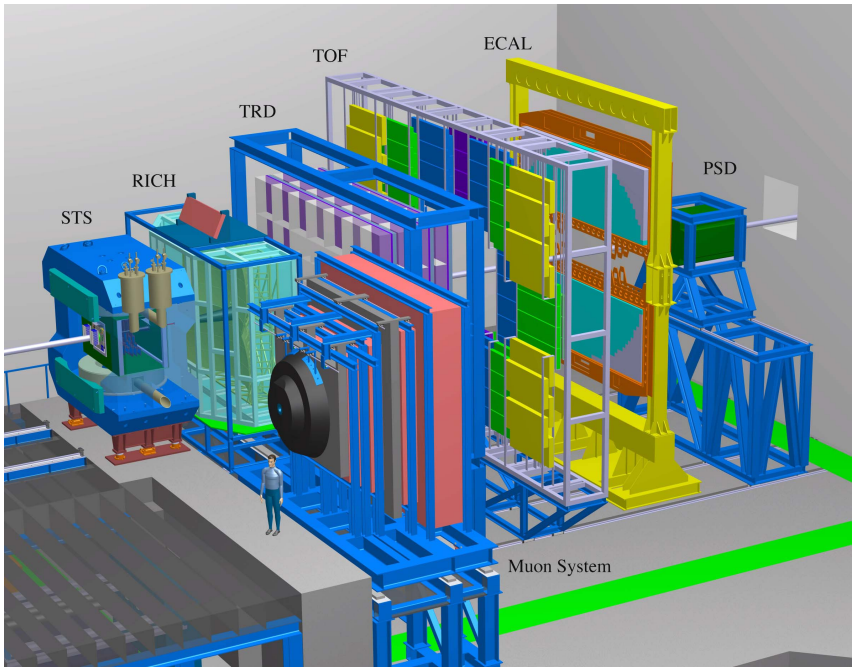

Figure 1. CBM setup for electron and hadron measurements. The beam enters from the left. Inside the dipole magnet, the target and the main tracking system (STS, MVD) are located. The downstream detectors comprise RICH and TRD for electron identification, TOF for hadron identification, ECAL for the measurement of neutral probes, and the PSD for event characterisation. The muon system is shown in its parking position; it can be interchanged on rails with the RICH detector.

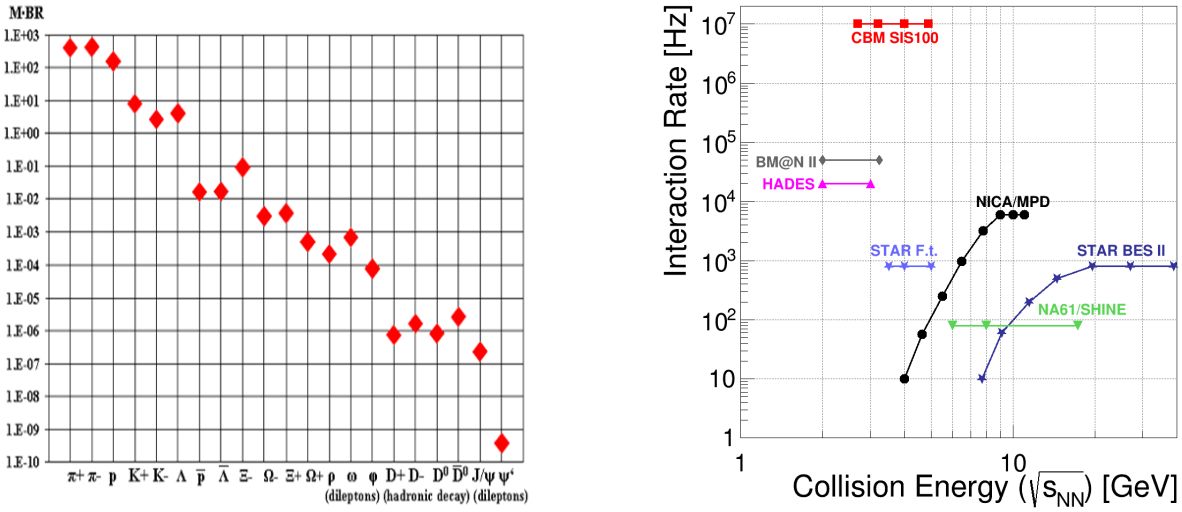

Figure 2. Left: model predictions for yields (multiplicity times branching ratio) of probes intended to be measured by CBM. Right: landscape of current or planned nuclear collision experiments at moderate collision energies, in terms of collision energy and maximal interaction rate (from [2]).

2. Readout concept

At the design interaction rate of 10 MHz, we expect a raw data rate of up to 1 TB/s from the detector front-end electronics. Current storage technologies allow archival rates of 100 Gbit/s or above; the limiting factor thus is not archival speed, but the cost of storage capacity. CBM hence requires the archival rate not to exceed 3 GB/s, which necessitates reducing the raw data rate online by a factor of 300 or more. The experimental challenge is to identify in real-time rare and complex decay topologies in a high track-density environment typical for heavy-ion reactions (figure 3), to select the corresponding data for archival and to discard the vast majority of the raw data. Because of the high interaction rates, which do not tolerate a significant trigger latency, and because of the complex trigger signatures which are almost impossible to implement in hardware, CBM chose a readout concept based on self-triggered, free-streaming front-end electronics, which deliver time-stamped data messages to the data acquisition system on activation of a read-out channel above a pre-defined threshold. The data acquisition aggregates and pushes all data to an online computing farm (FLES), where event reconstruction and selection are performed in software.

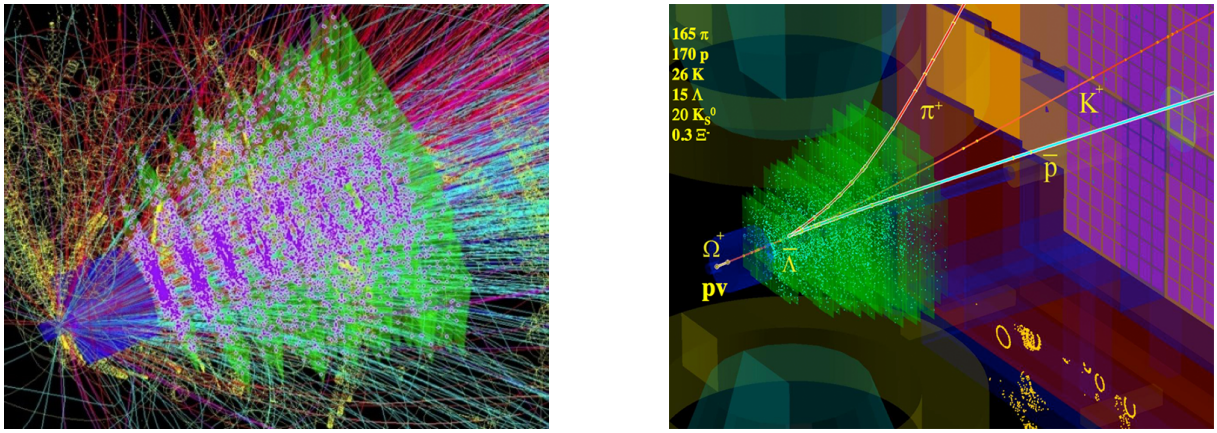

Figure 3. Left: A typical collision event (central Au+Au at 10A GeV) simulated in the CBM detector system. Right: Example of a trigger signature ($\overline{\Omega}^+ \to \overline{\Lambda}K^+ \to \pi^+\overline{p}K^+$) in the CBM detector.

On the FLES, two distinct software tasks will be executed. First, the data streams from the detector front-ends will be combined into "time slices", which represent the complete raw data information within a given time interval. This data aggregation is performed on the input nodes, the number of which is defined by the connectivity. Time slices are then delivered to the compute nodes, where reconstruction from raw data happens up to a level where the decision to keep or to reject data can be obtained. A description of the FLES and the time-slice building software can be found in [3].

3. Computing challenges

The interface of data acquisition and (online) reconstruction is the time slice, a container comprising the complete detector raw data within a time interval. The size of the time slice will be adjusted to the hardware architecture of the compute nodes. Typically, a time slice contains data from a large number of events (1000 or more). This concept has a number of consequences which impact the design of the reconstruction software:

- "Events" corresponding to physical beam-target interactions are not defined before reconstruction, e.g., by a hardware trigger, but have to be reconstructed from the data stream.

- At the maximal interaction rate of 10 MHz, the average time between subsequent events is 100 ns. The typical duration of one event is > 30 ns, the time-of-flight for $\beta = 1$ particles through the detector setup. Thus, many events will overlap in time.
- The problem of overlapping events is qualitatively different from the event pile-up in collider experiments, where events can be identified by reconstructing the interaction point along the beam axis in the beam overlap region. In our case, all interactions take place in the thin target (typically several 100 μm). A possible spatial separation can only be achieved in the coordinates transversal to the beam; the range of these coordinates is defined by the transverse beam profile, which is typically of a size of several mm.
- Many trigger signatures are complex. As an example, the selection of the decay $\overline{\Omega}^+ \rightarrow \overline{\Lambda}K^+ \rightarrow \pi^+\overline{p}K^+$ requires the reconstruction of two secondary vertices (Ω and Λ; see figure 3, right). This necessitates the complete reconstruction of the STS and the TOF detectors. In order to be applied in real-time, the reconstruction algorithms have to be very fast.

The required speed of the online reconstruction algorithms implies that one has to make use of massively parallel computing on both the data and the task level. The obvious first data-level parallelisation is possible due to the fact that time slices can be treated independently. Thus, many time slices can be processed in parallel on the FLES compute nodes, each node operating on one time slice at a time. Inter-communication between the compute nodes is not required. However, in order to make optimal use of the available computing power and to limit the size of the FLES farm to realistic figures in terms of investment, parallelisation is also required within each compute node.

Task-level parallelism within the reconstruction of one time slice is also obvious at the stage of local reconstruction (cluster and hit finding) between the different detector systems (STS, RICH, TOF etc.). Within a given detector system, data-level parallelism can be obtained by processing data from individual components simultaneously. For example, the STS consists of about 1000 sensors. Cluster and hit finding on each sensor is completely local, without interference of data from different sensors. Such reconstruction tasks will be distributed to the number of available cores within a compute node. Parallelisation at the track or vertex finding stage is less obvious, since here data from different detector parts have to be used as input. One indispensable key to high performance is vectorization of all reconstruction codes.

The development of corresponding reconstruction algorithms for CBM has started several years ago and is in a quite advanced state [4, 5, 6, 7]. As an example, figure 4 illustrates time-

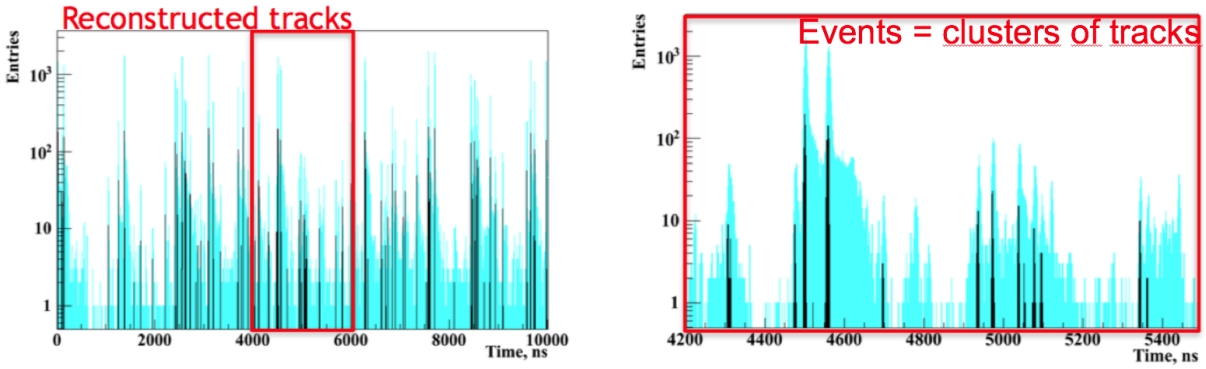

Figure 4. Time-based track finding in the STS (simulation of minimum-bias Au+Au events at 25A GeV beam energy and at 10 MHz interaction rate). Left panel: time of reconstructed tracks (black) on top of the input hits (light blue). Right panel: zoom of a region of the left panel. Clusters of tracks can be used to define events.

based track finding and event definition in the STS at 10 MHz interaction rate. Shown are the time distributions of the hits as input for tracking and that of the reconstructed tracks. On the hit level, separate events are visible, but it is also obvious that most events cannot be disentangled using the hit time information alone. The distribution of tracks, on the other hand, exhibitits narrow clusters which can be used to define events. The advantage of using tracks and not hits (or raw data) for the event definition is twofold: Firstly, since a track consists of a number of hits (up to eight), the track time resolution is smaller than the single-hit time resolution. Secondly, track finding implicitly discards late hits from secondary tracks, mostly spiraling delta electrons hitting the detectors several 100 ns after the interaction. The event reconstruction outlined above is able to correctly identify about 85% of all events. Work is ongoing to reliably tag the remaining 15% and either to discard those pile-up events or disentangle them by means of finding distinct primary vertices.

Software "triggers" build on reconstructed data as described above. A special software package was developed for the fast identification of decay patterns from reconstructed STS tracks [8]. A different approach was developed for the less complex charmonium signature. Here, data from the muon detection system alone can be used to arrive at a trigger decision. The computationally expensive reconstruction of the STS is thus not required. Our studies show that a trigger decision in this particular case can be obtained very fast, such that the amount of required compute nodes is rather small [9, 10, 11].

4. Simulation

For the development of proper reconstruction algorithms and for the assessment of the physics performance of the CBM detector system, in particular of the impact of the highest interaction rates on the latter, a realistic detector simulation is required. The detector simulation must model the free-streaming raw data output of front-end electronics and DAQ such as to provide a realistic input for the reconstruction. The input for the simulation is in general provided by some event generator for heavy-ion collisions, our most commonly used ones being UrQMD [12] and HSD/pHSD [13]. These generators deliver input in form of a list of primary particles for each physics event. It is the task of the simulation to destroy the association of particle hits to events, which should be re-established by the reconstruction routines. Our simulation comprises two stages: first, a transport engine (GEANT3 or GEANT4) is used to transport the primary particles through the detector setup, taking into account the relevant physics processes like decays, deflection in the magnetic field, interaction with the detector materials and creation of secondary particles. In order to be flexible with respect to the choice of the transport engine, we employ the Virtual Monte-Carlo features of ROOT as implemented in the FairRoot framework [14]. The particle transport is performed event by event, since no interference of tracks happens at this stage.

In a second step, the detector response simulation is performed. This simulation has to model the detailed physics processes in a given detector, leading to the registration of a signal (e.g., creation and propagation of electron-hole pairs in the silicon sensors), and the response of the read-out electronics – the "digitisation" in the strict sense. At this digitisation stage, interference of particles takes place if they cause charge to be delivered into the same read-out channel within the integration time of the read-out ASIC. In our case, such "double hits" can happen not only inside one event, but also for two particles from different events provided their time separation is small. Furthermore, a channel may be inactive for a certain time after having received a signal. The implementation of these behaviours requires buffering of signals for each read-out channel over a time interval corresponding to the integration or dead time of the electronics. In our framework, this is solved by the CbmReadoutBuffer class, which manages the data handling and buffering, providing an interface for the detector-specific treatment of double hits. Implementations of this time-based detector response simulation are available for

all CBM detectors relevant for the software triggers (STS, MUCH, and TOF) [15, 16].

The readout buffer instances send data to a software representation of the data acquisition, which aggregates data from different input sources, sorts them with respect to their time stamp and creates time slices. Since data from different detector systems can arrive with different delays, the DAQ emulation requires a second buffer stage to cope with data disordering. Internally, the DAQ buffer is realised by several STL maps with the time stamp as key, making use of the automatic sorting provided by this template container. The final output are TClonesArrays as branches of a TTree, each array representing raw data from a different sub-detector.

5. Software framework and event data model

The CBM software framework is built on top of FairRoot [14, 17], which uses ROOT as platform. In its original formulation, it provided a conventional, event-oriented framework with data arrays represented as branches of a TTree and one tree entry corresponding to one event. The schemes for simulation and reconstruction drafted in the previous sections show, however, that in order to describe the real experiment situation, the framework must be able to handle both event and time slice data simultaneously – the simulation converting input events into a data stream with individual data not associated to events, the reconstruction trying to reconstruct events back from this data stream in order to provide the suitable input for high-level physics analysis. First attempts to cope with this demands were taken several years ago by implementing CBM-specific extensions to the FairRoot framework [18]. Meanwhile, the FairRoot framework itself provides the necessary features, in particular for the simulation. Our event data model, however, had to be adapted to accommodate the real features of the CBM experiment.

The CBM data model continues to use TClonesArrays as tree branches, but now one tree entry corresponds to one time slice instead of one event. Events, once defined by some reconstruction task, are implemented as a collection of pointers / indices to data in their respective arrays. All tasks operating before the event definition have to operate on the entire time slice, i.e., in a "time-based" mode. The event-defining task has to register the data associated to one event to the event class. Subsequent tasks can make use of this event association to operate on data in an event-by-event mode. The scheme is illustrated in figure 5, comparing the conventional and the new event data model.

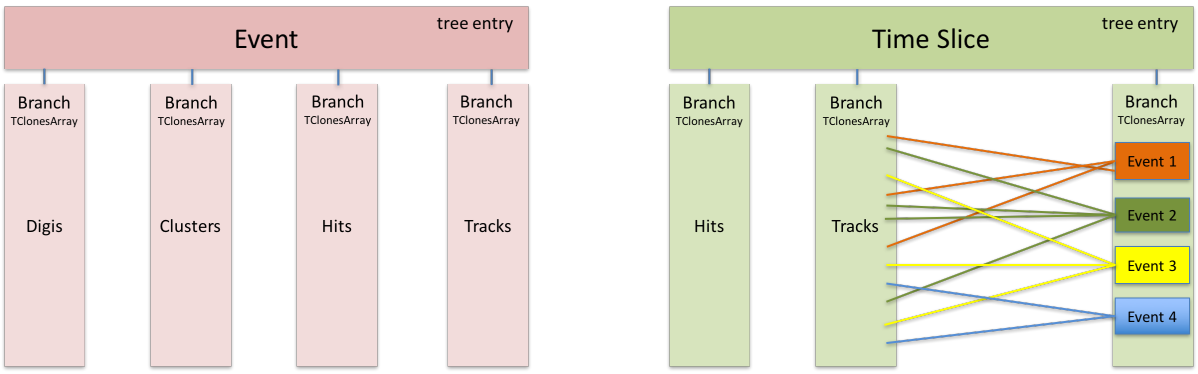

Figure 5. Event data model in CBM. Left: The "conventional" picture. Data are arranged in arrays as branches of a TTree; one tree entry corresponds to one event. Right: The current data model for CBM. Data are still in tree branches, but one tree entry corresponds to one time slice, comprising a large number of events. In this example, hit and track finding are performed before event definition; the event objects contain the indices of the tracks associated to an event.

The chosen approach comes with the penalty of one additional integer (index of data in its array) per data object, and a slightly more complicated loop over data in each event. This is a moderate price for the advantages that the data model offers:

- no data copy is required when associating a data object to an event;
- events can be defined on any data level (e.g., raw data ("digi"), hit or track);
- algorithms operating on the entire time slice ("time-based") as well as algorithms operating on events can be integrated in the same environment;
- the "ideal" case of event-by-event reconstruction can be described in the same data format (simulating one event per time slice only). This allows to isolate effects due to the interaction rate from others.

Simple performance checks demonstrated that the processing time per event for event-based reconstruction tasks does not depend on the number of events per time slice, i.e., that the computing overhead compared to the event-by-event case is negligible.

6. On- and offline computing

The raw data volume per typical CBM runtime (2 months per year) is estimated to be about 5 PB even at a moderate archival rate of 1 GB/s. The costs of storage media are potentially a limiting factor for the CBM physics output. The need to optimise the storage data volume and the fact that reconstruction from raw data can happen very fast (as the reconstruction algorithms must be suitable for real-time data processing) lead to the concept of archiving only raw data and not any type of reconstructed event data. Following this approach, offline analysis would always use reconstruction on-the-fly prior to high-level analysis. This suggests as storage data model a persistent time slice in the same format as the original time slice as delivered from the DAQ software, but skimmed from "uninteresting data" by online data processing. A consequence is that there is no formal difference between on- and offline algorithms. A base paradigm of CBM software is thus that algorithms used for online computing must be able to be used offline without modifications, i.e., there shall be a common environment for online and offline software.

The software framework described in the previous section serves the CBM needs for simulation and reconstruction of simulated data, i.e., for typical offline tasks. It is obviously also suited for offline reconstruction of real data if performance is not an issue. The natural question thus is in how far the framework can be applied for online purposes.

The often-heard opinion that the overhead connected to a ROOT-based framework is prohibitive for high-performance computing shall not be discussed here; of more importance is the fact that our current framework does not offer any concurrency model other than in-code vectorisation. The latest developments in the FairRoot framework, extending its functionality by a message-queue-based data transport layer [19] which provides asynchronous inter-process communication, constitute a very promising approach for the implementation of the developed data processing algorithms on parallel computing architectures. This approach will be investigated for CBM in the near future.

7. Summary

We have presented an overview of challenges for and approaches to computing for the CBM experiment. The data rates for CBM are comparable to those of LHC experiments after their high-luminosity upgrades. The particular features of CBM, namely the free-streaming data readout, call for a data processing framework and data model which go beyond the traditional event-based ones, enabling simulation of the free data stream and the online as well as offline deployment of reconstruction routines. Finding events and trigger signatures in a data stream

resulting from self-triggered front-end electronics constitutes a particular algorithmic challenge. The current status of the software developments suggests that the extreme interaction rates targeted by CBM are conceptually feasible. The most important task for the near future is the porting of the existing reconstruction routines to the new features of FairRoot which support concurrency.

References

[1] http://www.fair-center.eu/for-users/experiments/cbm.html
[2] Ablyazimov T et al (CBM collaboration) 2017 *Eur. Phys. J.* A **53** 60
[3] De Cuveland J and Lindenstruth V 2017 Online computing architecture for the CBM experiment at FAIR *this volume*
[4] Friese V 2012 *Springer Lect. Notes Comp. Sci.* **7125** 17
[5] Lebedev S et al 2014 *J. Phys.: Conf. Ser.* **513** 022019
[6] Kisel I 2015 *Eur. Phys. J. Web Conf.* **95** 01007
[7] Akishina V and Kisel I 2015 *J. Phys.: Conf. Ser.* **599** 012024
[8] Zyzak M 2016 *Online selection of short-lived particles on many-core computer architectures in the CBM experiment at FAIR* Dissertation, Goethe-Universität Frankfurt
[9] Singhal V et al 2012 *Proc. of the DAE Symp. on Nuclear Physics* **57** ed. S A Jain, P Shukla et al pp 972–973
[10] Singhal V, Chattopadhyay S and Friese V 2014 *CBM Progress Report 2013* ed V Friese and C Sturm (Darmstadt: GSI) p 104 (GSI-2014-00293, http://repository.gsi.de/record/64893)
[11] Ablyazimov T, Ivanov V and Friese V 2017 Finding the needle in the haystack: a charmonium trigger for the CBM experiment *this volume*
[12] Bleicher M et al 1999 *J. Phys.* G **25** 1859
[13] Cassing W and Bratkovskaya E L 2008 *Phys. Rev.* C **78** 034919
[14] Al-Turany M et al 2012 *J. Phys.: Conf. Ser.* **396** 022001
[15] Malygina H and Friese V 2017 A precise device need precise simulations: software description of the CBM Silicon Tracking System *this volume*
[16] Sitzmann P et al 2014 *J. Phys.: Conf. Ser.* **513** 022007
[17] http://fairroot.gsi.de
[18] Friese V 2011 *J. Phys.: Conf. Ser.* **331** 032008
[19] Al-Turany M et al 2014 *J. Phys.: Conf. Ser.* **513** 022001