The High-Rate Data Challenge: Computing for the CBM Experiment

Volker Friese
GSI Darmstadt

International Conference on Computing in High-Energy and Nuclear Physics
San Francisco, 10-14 October 2016
The Experiment

- **Compressed Baryonic Matter:** a heavy-ion experiment at the future facility FAIR in Darmstadt
- Fixed-target operation on extracted beams, 2 – 45 GeV/nucleon
- Spectrometer: silicon tracking system in a dipole magnetic field
- Hadron, lepton and photon ID: RICH, Muon System, TRD, TOF, ECAL
- Observables: yields, spectra, flow, correlations, fluctuations of bulk hadrons, multi-strange hyperons, open charm and charmonium; low-mass di-leptons
- First beam in 2022
Characteristics

- Versatility: exchange or replace detector systems according to physics aim (e.g. electrons / muons) or conditions (beam energy)

- Complexity: up to 600 charged tracks per collision in the acceptance

- Capability: up to $10^7$ collisions per second
The Rate Problem

• CBM targets at extremely rare probes, which necessitates very high interaction rates (design rate 10 MHz).
• That entails a raw data rate of up to 1 TB/s.
• To be reduced online to a storage rate of several GB/s.
• Trigger signatures are mostly complex (e.g. weak cascade decays) and cannot be realized in hardware.
• Readout concept:
  – No hardware trigger
  – Self-triggered front-end electronics deliver time-stamped data
  – Data-push architecture to online compute farm
  – Event reconstruction and selection to be performed on CPU
Online Data Flow

- Data are aggregated and pre-processed in an FPGA layer near the experiment.
- Time-slice building is performed on CPU (input nodes).
- Event reconstruction and selection is performed in real-time on CPU (compute nodes) in the GSI "Green Cube".
Consequences for Online Computing

• Reconstruction does not start from events (defined by hardware trigger) but from „time slices“ containing many events.
  – size of time slice adjusted to architecture of compute farm
  – typical value: 100 MB (1000 events)
  – one time slice delivered to one compute node; avoid intercommunication between compute nodes
  – events can overlap in time; no trivial event definition: ”4-D reconstruction”

• All online algorithms have to be extremely fast
  – Trivial data-level parallelism for time slices (one time slice per node)
  – Use massive parallelisation also within one node (many-core CPU/GPU/...)

Volker Friese
CHEP2016, San Francisco, 11 October 2016
Parallelisation Within a Time Slice

- MVD hit finding
- STS cluster finding
- RICH ring finding
- TRD hit finding
- TOF hit finding
- ECAL cluster finding

- STS hit finding
- TRD track finding
- Global Tracking
- Vertexing, PID

Volker Friese
CHEP2016, San Francisco, 11 October 2016
Parallelisation Within a Time Slice

Task Level Parallelism

- MVD hit finding
- STS cluster finding
- RICH ring finding
- TRD hit finding
- TOF hit finding
- ECAL cluster finding

STS hit finding

STS track finding

TRD track finding

Global Tracking

Vertexing, PID
Parallelisation Within a Time Slice

Task Level Parallelism

STS hit finding
STS track finding
STS track fitting

Data Level Parallelism

MVD hit finding
STS cluster finding
RICH ring finding
TRD hit finding
TOF hit finding
ECAL cluster finding

> 1000 sensors

Volker Friese
Example: CA Track Finder

Given n threads each filled with 1000 events, run them on specified n cores, thread/core. Track finding is performed on a stream of hits. Events can be defined based on found tracks.

Good scalability of algorithm with multi-threading on many core CPUs.
Framework and Data Model

• CBM uses the FairRoot framework (built on ROOT) for simulation, reconstruction and analysis.
• The data model is based on the ROOT TTree.
  – Different data branches: raw data (digis), clusters, hits, tracks, vertices, ...
  – A “run” produces an output tree from an input tree
• Conventionally, one tree entry corresponds to one event (collision)
• We have to deal with both time slices and events
  – In simulation: convert events (Monte-Carlo) into time slices (destroy association of data to events)
  – In reconstruction: reconstruct events from time slices
• Situation when output tree entry does not correspond to input tree entry not mapped in the framework
- No data copy when associating data to event
- Small overhead (one pointer/index per data object)
- Events can be defined based on any data level
- Algorithms are flexible to run on entire time slice (4-d reco) or on defined events (analysis)
- Ideal case (event-by-event) described in the same format (one event per time slice)
Outlook: Offline Computing

• Raw data volume per typical runtime (2 months): about 5 PB
• Limiting factor will not be computing capacity but storage costs
• Ansatz: store only raw data
  – For offline analysis: reconstruct on-the-fly
  – Assumes fast online algorithms deliver close-to-final precision
• Storage model is time slice with raw data, skimmed online from “uninteresting” data
• Consequence: no formal difference of online and offline algorithms
  – Use same framework
• But: no support of concurrency in the current ROOT-based framework
Outlook: A Concurrency Framework

- FairMQ: extension of FairRoot with a message queue-based data transport framework, providing asynchronous inter-process communication
- Promises flexibility w.r.t. architecture and data model
- Will be explored by CBM in the near future
Summary: Computing Challenges for CBM

- Huge interaction and data rates necessitate real-time event reconstruction and data selection
  - Reduce about 1 TB/s to several GB/s in real time in software
- Basis of the data model is a time slice containing many events
- Fast 4-D reconstruction algorithms under developments
  - Many achievements, but still some way to go
- Quest for a common online and offline software framework
  - Concurrency needed
  - Common data model allowing time-based and event-based analysis without change of code
  - Make use of the extension of the current FairRoot to FairMQ