Core software aspects for the LHCb upgrade

Niklas Nolte on behalf of the LHCb collaboration
ACAT 2019 - Track 1 - 12.03.2019
Introduction

➢ Major LHCb detector upgrade
  i. New detector components
  ii. Removal of hardware trigger stage “L0” → 30 MHz of pp collisions to be processed by a software trigger (HLT)

➢ The framework and the trigger need to have greatly improved performance
LHCb software

- Development started in early 2000s, some code even older
- Based on the software framework “Gaudi”
  - 1. shared with ATLAS and others
  - 2. Was initially not laid out for MultiThreading (MT), only process forking → weak scalability in RAM usage
  - 3. Now: MT in “GaudiHive” as part of Gaudi
  - 4. Gaudi::Functional: MT friendly by construction
- LHCb software: reconstruction + selection algorithms are ported to Gaudi::Functional
Gaudi::Functional

➢ Functional stateless wrapper around the base class “Algorithm”

➢ Algorithms store and load their data from an event store, Gaudi::Functional handles this for the user

➢ Inputs and Outputs are defined before processing
→ Easier to trace data dependencies

➢ User only implements the operator() const → MT friendly and reentrant
Event Store

➢ Each event has own event store
➢ Data are identified by string “/some/location/data”
➢ Implemented as tree structure
➢ Retrieving elements is slow
  ➢ tree is traversed
  ➢ lookup in a map for each step

➢ New implementation (not yet fully integrated):
  ➢ Hashmap { full locations → data }, no more tree structure
  ➢ Speeds up the lookup process greatly
Data Layout

➢ Earlier: collection of pointers “KeyedContainer<Track>”, essentially std::vector<Track*> with much additional logic
   → Many small memory allocations
   → Much pointer chasing

➢ Now: std::vector<Track>
   → Preallocate as good as possible, larger allocations, but fewer
   → Still not the most efficient layout, although much better already
# Removal of KeyedContainer

<table>
<thead>
<tr>
<th>Function</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator new</td>
<td>16.7%</td>
</tr>
<tr>
<td>_int_free</td>
<td>8.3%</td>
</tr>
<tr>
<td>PrPixelTracking::bestHit</td>
<td>5.7%</td>
</tr>
<tr>
<td>PrForwardTool::collectAllXHits</td>
<td>5.7%</td>
</tr>
<tr>
<td>PrStoreFTHit::storeHits</td>
<td>4.5%</td>
</tr>
<tr>
<td>PVSeed3DTool::getSeeds</td>
<td>2.7%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrPixelTracking::bestHit</td>
<td>19.2%</td>
</tr>
<tr>
<td>PrForwardTool::collectAllXHits</td>
<td>9.9%</td>
</tr>
<tr>
<td>operator new</td>
<td>8.9%</td>
</tr>
<tr>
<td>PrStoreFTHit::storeHits</td>
<td>7.9%</td>
</tr>
<tr>
<td>PVSeed3DTool::getSeeds</td>
<td>5.5%</td>
</tr>
<tr>
<td>_int_free</td>
<td>5.1%</td>
</tr>
</tbody>
</table>

Top 6 CPU consumers in the upgrade HLT1 stage before (left) and after (right) the removal of KeyedContainer
Investigation ongoing:
switch to a SoA data layout as opposed to current AoS approach

Possibly wrap into “SOAContainer”
→ merges advantages of AoS and SoA
  i. allows underlying SoA structure
  ii. still able to ask for the underlying scalar structure

1: https://indico.cern.ch/event/736105/contributions/3036391/attachments/1687650/2717271/talk.pdf
Scheduling Algorithms

➢ First MT scheduler in Gaudi: AvalancheScheduler
  i. Enables granular levels of parallelisation (within Events, or even within Algorithms)
  ii. Too complex and much too slow for LHCb Run III trigger use case

➢ New scheduler with following goals:
  i. Minimal runtime overhead
  ii. Support arbitrary control & data flow
  iii. Parallelize over events (1 event/task)
Trigger control flow

Decision

Line_1

P_1 → G → F_1

Line_2

P_2 → G → F_2
Control flow nodes

➢ Composite Nodes
  i. Logic (AND | OR | NOT)
  ii. Children
  iii. Execution policy: “allow short-circuiting” or “execute all children”
  iv. Ordering constraint on children (control flow edges)

➢ Basic Nodes
  i. Manage one algorithm
  ii. List of data dependencies
  iii. Distributes decision provided by the underlying algorithm
The user configures:

i. control flow via a set of composite and basic nodes
ii. data flow by defining algorithm inputs and outputs

Construct data dependency list for each basic node by matching inputs and outputs.

Order basic nodes into flat list respecting ordering constraints.
Ordered execution

Execution Order

P₁ → P₂ → G → F₁ → F₂

Decision

Line_1

Line_2
Runtime

Execution:

- Go through ordered list of basic nodes
- If node requested by any parent node, i.e. the parent is not yet evaluated...
  a. Execute all needed data producers if not yet executed
  b. Execute Basic Node
  c. Notify parents about decision
  d. Parents evaluate if execution policy permits it
Scheduling algorithms - technical aspects

➢ Master thread prepares self-contained tasks (full events) for a threadpool → no thread synchronization

➢ Performance overhead:
  i. no measurable overhead in the upgrade HLT1 reconstruction
  ii. less than 1% overhead in a mock HLT1 with 20 trigger lines processing at 30 kHz per computing node
  iii. less than 2% on a mock HLT2 with 1000 trigger lines
Further ongoing work

➢ Optimization of reconstruction algorithms

➢ DD4Hep as detector description (DD) is being integrated
  i. Current DD is quite outdated and lacks some features
  ii. Better maintainability because DD4Hep is not LHCb-exclusive

➢ Adaptation and optimization of the “conditions database” (time-varying conditions) and the access to it
  i. Make it compatible with the upgrade code
  ii. Speed up the framework startup time
Thanks
Barrier - share the work

➢ Scenario: Multiple algorithms select intersecting subsets of the same collection
➢ A following, expensive algorithm can be invoked once on the union of all selections → avoid duplication of work
➢ This requires optional data dependencies
➢ Imposes additional order constraints on basic nodes
LHCb Trigger Run II vs Run III

LHCb 2015 Trigger Diagram

40 MHz bunch crossing rate

L0 Hardware Trigger: 1 MHz readout, high $E_T/P_T$ signatures

450 kHz $h^z$
400 kHz $\mu/\mu$
150 kHz $e/\gamma$

Software High Level Trigger

Partial event reconstruction, select displaced tracks/vertices and dimuons

Buffer events to disk, perform online detector calibration and alignment

Full offline-like event selection, mixture of inclusive and exclusive triggers

12.5 kHz (0.6 GB/s) to storage

LHCb Upgrade Trigger Diagram

30 MHz inelastic event rate (full rate event building)

Software High Level Trigger

Full event reconstruction, inclusive and exclusive kinematic/geometric selections

Buffer events to disk, perform online detector calibration and alignment

Add offline precision particle identification and track quality information to selections

Output full event information for inclusive triggers, trigger candidates and related primary vertices for exclusive triggers

10 GB/s to storage
References

➢ LHCb collaboration, Technical design report Trigger, March 2018, CERN-LHCC-2018-007. LHC-B-TDR-017

➢ LHCb Collaboration, Trigger schemes website, visited 24.02.2019, modified 2-5 Gb → 10 Gb

➢ Manuel Schiller, “SoAContainer”, visited 25.02.2019,
https://indico.cern.ch/event/736105/contributions/3036391/attachments/1687650/2717271/talk.pdf

➢ Illya Shapoval, Adaptive Scheduling Applied to Non-Deterministic Networks of Heterogeneous Tasks for Peak Throughput in Concurrent Gaudi, CERN-THESIS-2016-028