Machine learning and parallelism in the reconstruction of LHCb and its upgrade

Marian Stahl*, on behalf of the LHCb collaboration
(marian.stahl@cern.ch)
*Heidelberg University

22nd International Conference on Computing in High Energy and Nuclear Physics
October 13th, 2016
- General purpose single arm forward spectrometer at the LHC
- Broad physics programme with focus on studying $CP$ violation in beauty and charm decays and rare decays of $b$ and $c$ hadrons
- Excellent detector performance
  - $\langle \varepsilon_{\text{track reconstruction}} \rangle > 95\%$, $\frac{dp}{p} \sim 0.5 - 1\%$, decay time resolution $\sim 45$ fs
Main track types for physics analyses

- **Long tracks** Hits in VErtexLOcator, InnerTracker or/and OuterTracker (and mostly TriggerTracker). Used in majority of analyses
- **Downstream tracks** Hits in TT and IT/OT. Tracks from daughters of long lived particles (e.g. $\Lambda$, $K_0^0$)

Two phases in LHCb tracking: Track finding / pattern recognition and track fitting (Kalman filter)

Main challenges: **Fast and efficient** reconstruction at **low fake rate**

- Parallelization
- Machine Learning
Moved to real time reconstruction, alignment and calibration set-up in Run II

Need same reconstruction online and offline

Track reconstruction in two stages

- Fast stage (HLT1) for long tracks with $p_T > 500$ MeV and tighter track quality requirements
- Full stage (HLT2) achieves offline efficiency and precision (details in backup)

**LHCb 2015 Trigger Diagram**

- 40 MHz bunch crossing rate
- L0 Hardware Trigger: 1 MHz readout, high $E_T/P_T$ signatures
- $450 \text{ kHz } h^\pm$, $400 \text{ kHz } \mu/\mu\mu$, $150 \text{ kHz } e/\gamma$
- 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
Parallelization

Parallelization

Parallelization
Parallelization challenges: (here: SIMD)

- Reconstruction code: selection (branching) rather than identical computation
  ⇝ Need to find time-consuming hot spots for SIMD
Parallelization challenges: (here: SIMD)

- Reconstruction code: selection (branching) rather than identical computation
  ➞ Need to find time-consuming hot spots for SIMD

- Forward tracking starts from VELO seed tracks. Propagate track path through $B$ field and project T station hits for each seed into Hough plane
  - Ideal $B$-field: two straight lines with intersection in middle of magnet
  - Real world: describe track path through fringe field with 3$^{rd}$ order polynomial

- SIMD: Calculate Hough projection of two hits in parallel (SSE: 128 bit width = 2 doubles)
  ➞ Speedup of 40 %

SIMD in forward tracking
- Track fitting is done with Kalman filter
- Largest timing contribution in HLT1 due to expensive calculus (Runge-Kutta for field equations, $5 \times 5$ matrix operations)
- SIMD for transportation of covariance matrix from state $k \to k + 1$:
  Transform $F \cdot C \cdot F^T$ (corresponding vectorized function used at several places)

⇒ Once data "in right place" (AVX):
  7 instructions (4 if FMA x86 available) for 16 '*' and 3 '+'
～ Gain factor
  ~ 2 by writing out calculation
  ~ 5 by using AVX
  (on average ~ 2 FLOPs/cycle)

- Ongoing investigations:
  See Daniel Campora Perez' talk
- Fake tracks mainly from wrong matches between VELO and T station. No detector in the magnet, very long lever arm.
- Remaining fake tracks from Kalman-filter $\chi^2$/dof cut $\approx 22\%$
- $\Rightarrow$ Improved to $\approx 14\%$ using neural network (NN, type: MLP). It's output is called *ghost probability*

**Timeline:**
- Processed offline $\rightarrow$ Used in HLT2 $\rightarrow$ Used in HLT1

(2015) Speedup by factor $\sim 90$; Combined with Kalman-filter $\chi^2$/dof; Increases Downstream track eff.
(2016) Reduces HLT2 combinatorics by 40%; Negligible efficiency loss
Want to reject fake tracks at stable efficiency in early stages of processing, namely in **forward tracking** (new in 2016)

- Search window in T stations defined by VELO track estimate and minimal $p_T$
- Project x-hits into a reference plane $\Rightarrow$ clusters
- Fit x-clusters and remove outliers
- Add and fit track with stereo hits $\rightarrow$ Kalman filter
- Recovery loop in HLT2 for track candidates with hits in only 4 x-layers

**Trained two neural networks (MLPs)**

- For rejection of bad 4-layer-x-clusters in recovery loop
- For track candidate selection after stereo fit (HLT1 & HLT2)
• NNs trained for background rejection at given (97 to 99 %) efficiency
  • Hidden Layer (HL) architecture most important hyperparameter
    ⇝ NN in recovery loop (RL): 9 Input nodes, 16,10 HL nodes
    ⇝ NN after stereo fit: 16 Input nodes, 17,9,5 HL nodes

• NN responses & other parameters tuned with MC and minimum bias data

<table>
<thead>
<tr>
<th>MC performance</th>
<th>$\nu = 1.6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016 w.r.t. 2015</td>
<td>w/ RL</td>
</tr>
<tr>
<td>timing HLT1</td>
<td>±0 %</td>
</tr>
<tr>
<td>timing HLT2</td>
<td>+4 %</td>
</tr>
<tr>
<td>fake rate</td>
<td>−27 %</td>
</tr>
<tr>
<td>fake rate HLT1</td>
<td>−15 %</td>
</tr>
<tr>
<td>$\varepsilon$ long</td>
<td>+0.5 %</td>
</tr>
<tr>
<td>$\varepsilon$ long from B</td>
<td>+0.2 %</td>
</tr>
<tr>
<td>$\varepsilon$ HLT1 long from B $p &gt; 3, p_T &gt; 0.5$ GeV</td>
<td>+0.1 %</td>
</tr>
</tbody>
</table>

• Results:
  • Increased efficiency
  • Reduced fake rate considerably
  • Decreased speed compensated in later stages due to fake track rejection
  • NNs only contribute 2 % (HLT2), 0.5 % (HLT1) to timing of forward tracking algorithm
Upgrade design luminosity forces to move to hardware-triggerless readout at 30 MHz

All tracking detectors will be upgraded
  - Higher reconstruction efficiency at higher occupancy

Only \(\approx 15\) ms for fast trigger stage / event
  compare to 40 ms (800 ms) in HLT1 (HLT2) in Run II

Dedicated upgrade studies:
  - GPGPU usage for VELO tracking
  - Cellular automaton tracking for T station standalone tracks (also w/ GPUs?)
  - Need to optimize performance/cost

Most improvements from Run II portable to upgrade tracking
The time consumption of the LHCb reconstruction software was reduced by a factor 2 from 2012 to 2015 thanks to the help of parallelization in hot-spots of the software.

There is continuous effort to reduce time consumption further, especially in view of the upgrade.

With the help of machine learning, LHCb reduced its rate of fake tracks by about 40% in the trigger:
- By a neural network after track fitting
- By two neural networks in the forward tracking (new in 2016)

The upgrade of LHCb will use a purely software-based trigger:
- This poses severe restrictions on the timing-budget
- Parallelization will be crucial for upgrade track reconstruction
Backup slides start here
HLT1
- Reconstruct all VELO tracks
- Use simplified Kalman filter to fit VELO tracks ⇒ reconstruct PV
- Propagate VELO tracks to TT and get charge estimate
- Propagate VELO tracks to T stations, select $p_T > 500$ MeV. Find matching hits ⇝ long track candidate
- Fit long tracks with Kalman filter

HLT2
- Propagate unused VELO tracks to T stations, find all long tracks
- Reconstruct T-station standalone tracks and downstream tracks
- Fit with Kalman filter

\[ T = 11.6 + 23.4/p \quad \text{2012 Data: } T = 11.8 + 23.7/p \quad \text{2015 25 ns Data:} \]
Need for speed: parallelization

- Trigger output rates: \((0.2 \rightarrow 5.0 \rightarrow 12.5)\) kHz
  but it could still be more!

- Track reconstruction is main time consumer
  \(\rightsquigarrow\) exploit advances in parallelization technology

- Not all of our hardware is identical: need set of instructions that works on all CPUs
  \(\rightsquigarrow\) common standard Intel SSE2

- Concentrate on SingleInstructionMultipleData in reconstruction tasks on multicore CPUs

- Problems: SIMD needs to be put in by hand. Reconstruction software contains a lot of selection (branching), not so much identical computation
  \(\rightsquigarrow\) Need to find time-consuming hot spots for SIMD
Speedups necessary to run fake track rejection in trigger. Speedup of $\mathcal{O}(\times 90)$ reached from Run I to Run II. A general list:

- applying neural network faster than BDT ($\times 40$)
- compile network instead of loading at runtime ($\times 4$)
- tune autogenerated network code ($\times 2$)
- Rectified Linear Unit activation function ($\times 4$)
- AVX parallelization ($\times 10$)

Ideas and plans

- switch to SSE3/AVX
- deep learning/human assisted variables
- "domain adaptation" [arXiv:1409.7495 [stat.ML]] tried, but unsuccessful
Parallelization and machine learning play major role in timing improvements
On the other hand: revisiting the algorithms and making smarter choices helped to improve timing at many points
The result: an overall speedup of about a factor 2
Buying CPUs to have the same physics performance: $O(MCHF)$