Fast Kalman Filtering: new approaches for the LHCb upgrade

CHEP 2018, Sofia, Bulgaria

Plácido Fernández Declara on behalf of the LHCb collaboration

July 10, 2018

CERN
1. LHCb Upgrade and Kalman filter

2. Vectorized Kalman filter

3. Parametrized Kalman filter

4. Further simplifications

5. Conclusions
LHCb Upgrade and Kalman filter
The LHCb Upgrade

- Run at higher luminosity
  \[ 4 \cdot 10^{32} \text{cm}^{-2}\text{s}^{-1} \text{ (Run I,II)} \rightarrow 2 \cdot 10^{33} \text{cm}^{-2}\text{s}^{-1} \text{ (Run III)} \]
- Upgrade to full software trigger:
  - **From**: L0 hardware trigger
    \[(30MHz \rightarrow 1MHz)\]
  - **To**: 30MHz detector readout
- Upgraded tracking subdetectors:
  VELO, UT and SciFi
Fast Kalman filter

Track reconstruction:

- Reconstruct VELO tracks.
- Add the UT hits.
- Find matching hits in SciFi.

Used to obtain an optimal track estimate, the Kalman filter is applied in both the ”fast” stage to select tracks, and the ”best” stage to give ultimate momentum resolution.

Depending on the complexity of the Kalman fit which is performed, it can contribute up to 60% of the ”best” sequence time.
Kalman filter at LHCb

- Well-known quadratic estimator, where for every hit we "predict" and "update" the state according to the model and the measurements.
- 3 steps: forward filtering, backwards filtering and smoother.
- High volume of small matrix operations.
- Not trivial to be parallelized.
Vectorized Kalman filter
Vectorized implementation

- Using SIMD, various filter steps are calculated for N tracks, in parallel
- Maximize Vector units usage. (Tracks have different number of hits)

Scheduler
  - Use of static scheduler for available cores and vector processing units
  - The scheduling applies to all steps (forward, backward and smoother)

Data layout
  - AOSOA: Array Of Structure Of Array
  - Benefit from both SIMD and cache
  - Adapt to vector width in compile time (cross-architecture)

- Precision can be changed between single and double to test stability of the calculations, and exploit different hardware.
Cross-architecture Kalman fit - Throughput

Throughput (fit and smoother / s)

Active processors

1 2 4 8 16 32 64 128 256

2x Intel Xeon® CPU E5-2630 v3
2x Intel Xeon® CPU E5-2630 v4
Intel Xeon Phi™ CPU 7210
2x PowerNV 8335-GCA
Cavium ThunderX Dual 48 Core (ARM64)
2x AMD EPYC 7351

[Cámpora Pérez e.4483]
Cross-architecture Kalman fit - Roofline

Arithmetic intensity [FLOP/Byte]

Performance [GFLOP/s]

Intel Xeon® E5-2630 v3
AMD EPYC 7351
Intel Xeon® CPU E5-2630 v3
NVIDIA TESLA P100 Pascal
NVIDIA GTX TITAN X Maxwell

[Cámpora Pérez e.4483]
Parametrized Kalman filter
Parametrized Kalman filter

- The slow parts of the Kalman filter are:
  - The extrapolation through the magnetic field
  - The magnetic field and the material look up
- We replace this parts with parametrizations for the extrapolations between layers in the detector.
  - We apply "simple" functions outside the magnet region, and more complex functions inside it.
  - Extrapolation from one detector layer to the next is done with functions that map the state at position \( z \) to a state at position \( z' \)
  - The magnetic look up is not necessary since each detector layer has its own tuned parametrized extrapolation.
  - Material effects are modelled for every extrapolation function with a noise matrix added to the state covariance matrix. Energy-loss is not directly modelled.
First hit in the VELO - long tracks
Parametrized Kalman filter - Momentum resolution

LHCb simulation

\[ \sigma_p \]

LHCb simulation

\[ \sigma_p \]

LHCb twiki
Further simplifications
Further simplifications

For the parametrized Kalman filter:

- A new version of the parametrized Kalman allows to cover the discrepancies for low momentum resolution, and the larger angle in X.
- Being tested, coming soon.

Grouping measurements:

- For the tracking stations the measurements could be grouped, processing a smaller number of nodes.
- To be tested, but this could simplify the computations for faster processing.
Information Filter

- Expressing it with the inverse covariance matrix:
  - \( W = P_{k|k-1}^{-1} \)
  - \( t = W \cdot x_{k|k-1} \)

- Simplification of some matrix operations, e.g. noise step can be done with an approximation using only the terms \((t_x, t_x)\) and \((t_y, t_y)\).

- There is no need for an artificial covariance matrix at the beginning.

- This should allow to run in single precision, thus increasing the performance when computing.

- There are some challenges to solve with the new formulation.
  - e.g. Inversion of non symmetric 5x5 matrix.

- This is an ongoing work, still not tested in the framework.
Conclusions
Conclusions

- Vector implementation: great performance on different architectures thanks to data layout and scheduler. 10%-20% performance gain.
  - Integrated in Gaudi framework and ready to use.
- Parametrization:
  - extrapolation/material requires 30%-50%. Simplified parametrization can speed up by a factor 5-10.
  - We can predict which tracks will give us comparable results to the full Kalman filter.
- Further simplifications could yield better results in the parametrizations.
- Moving to an Information filter could allow to compute in single precision and apply other simplifications, with the potential performance gain.
Questions?