Parallelized Kalman-Filter-Based Reconstruction of Particle Tracks on Many-Core Architectures with the CMS Detector

G Cerati[4], P Elmer[2], B Gravelle[5], M Kortelainen[4], V Krutelyov[1], S Lantz[3], M Masciovecchio[1], K McDermott[3], B Norris[5], A Reinsvold Hall[4], D Riley[3], M Tadel[1], P Wittich[3], F Würthwein[1] and A Yagil[1]

[1] University of California, San Diego, La Jolla, CA, USA 92093
[2] Princeton University, Princeton, NJ, USA 08544
[3] Cornell University, Ithaca, NY, USA 14853
[4] Fermilab, Batavia, IL, USA 60510-5011
[5] University of Oregon, Eugene, OR, USA 97403

E-mail: `mario.masciovecchio@cern.ch`

Abstract. In the High–Luminosity Large Hadron Collider (HL–LHC), one of the most challenging computational problems is expected to be finding and fitting charged-particle tracks during event reconstruction. The methods currently in use at the LHC are based on the Kalman filter. Such methods have shown to be robust and to provide good physics performance, both in the trigger and offline. In order to improve computational performance, we explored Kalman-filter-based methods for track finding and fitting, adapted for many-core SIMD (single instruction, multiple data) and SIMT (single instruction, multiple thread) architectures. Our adapted Kalman-filter-based software has obtained significant parallel speedups using such processors, e.g., Intel Xeon Phi, Intel Xeon SP (Scalable Processors) and (to a limited degree) NVIDIA GPUs. Recently, an effort has started towards the integration of our software into the CMS software framework, in view of its exploitation for the Run III of the LHC. Prior reports have shown that our software allows in fact for some significant improvements over the existing framework in terms of computational performance with comparable physics performance, even when applied to realistic detector configurations and event complexity. Here, we demonstrate that in such conditions physics performance can be further improved with respect to our prior reports, while retaining the improvements in computational performance, by making use of the knowledge of the detector and its geometry.

1. Introduction

Finding and fitting charged-particle tracks is one of the most computationally challenging steps of the event reconstruction in the CMS [1] detector. For the online reconstruction, it has a direct impact on the rate of data recorded by the CMS High Level Trigger (HLT), which are selected from the up to 100 kHz (up to 750 kHz in the High-Luminosity LHC, HL–LHC, era) CMS Level–1 trigger acceptance rate. For the offline reconstruction, it is a limitation on the amount of data that can be processed for physics analyses. The computational challenge represented by the track finding and fitting will become even more important in the HL–LHC era, due to the increase in the instantaneous luminosity and in pileup (PU) of primary interactions per event, leading to a significant increase in the track reconstruction time.

With processors of all types gaining more and more of their performance from highly parallel elements, the LHC experiments must exploit this feature in order to sustain the higher HL–LHC processing requirements. We focus on the traditional Kalman Filter (KF) method [2], and adapt it to efficiently exploit highly parallel architectures, such as Intel Xeon Phi, Intel Xeon SP (Scalable Processors), and NVIDIA GPGPUs. In this context, we have developed the mkFit program, with the ultimate goal of reaching physics performance comparable with the current CMS track reconstruction [3], while achieving a significant improvement in computational performance.

2. Parallelized Kalman Filter tracking

The mkFit project started in 2014, with the development of a Matriplex matrix operation library, optimized for the simultaneous vectorized processing of sets of small matrices. Relying on this library, the initial implementation of a vectorized KF track fitting algorithm was demonstrated in a simplified detector geometry [4], followed by the initial implementation of an analogous track finding algorithm [5]. Further developments [6–8] allowed the software to achieve satisfactory performance, using Intel Threading Building Blocks (TBB) parallel constructs to comply with the CMS software (CMSSW) code base. A first implementation of the mkFit program on GPGPUs was also pursued [9]: while the Matriplex library was found to outperform standard small-matrix multiplication packages for GPUs, the performance of the KF-based track finding on GPGPUs is as of today not satisfactory.

After demonstrating satisfactory performance using a simplified detector geometry, the mkFit program was extended to handle a realistic detector geometry, as well as to retain performance in realistic high detector occupancy scenarios, such as the high PU scenarios expected at the HL–LHC.

The latest developments, whose effects are illustrated in this document, aim at further improving the mkFit physics performance through the optimization of hit and track selection algorithms, as well as at integrating the program with CMSSW, with the goal of integration in the CMS HLT test-bed system for Run III of the LHC. Future work will focus on the implementation of the Phase–II CMS geometry, with the ultimate goal of employing mkFit in the CMS HLT, and possibly offline, during the HL–LHC era.

3. The CMS detector geometry and events

This document illustrates the performance of mkFit for track reconstruction in simulated $t\bar{t}$ events with PU of 70 (50), using the Phase–I CMS detector geometry, that reflect the expected (actual) data taking conditions during the LHC Run III (Run II). Two operation modes are available: standalone, where mkFit operates independently of CMSSW; and within CMSSW.

3.1. Geometry and detector description

In the mkFit program, geometry is described as a vector of `LayerInfo` data structures containing: physical dimensions of a layer; hit search windows; and auxiliary parameters and flags for track finding. The last category includes information about layer detector type, individual detector module structure (double-sided, *mono+stereo*, or single-sided, *mono*), and holes in detector coverage (currently only used for the CMS endcap detectors).

In order for the track finding algorithm to be agnostic of the actual detector structure, *tracking regions* are defined together with the corresponding *steering parameters*. In the current program, five distinct regions in η are defined (barrel, $+z/-z$ transition, $+z/-z$ endcap). Tracking regions could as well be defined according to track p_{T} in future development. The steering parameters consist of a vector of `LayerControl` data structures, containing indices of the layers to be traversed during track finding, as well as layer parameters and flags that are specific to a tracking region (e.g., information about potential seeding layers). As a result, the track finding algorithm simply follows the layer-to-layer propagation defined by the steering parameters, executing operations in accordance with the control flags contained in the `LayerInfo` and `LayerControl` data structures.

The setup described in this section is implemented as a plugin. Hence, it is possible to support any detector geometry. In the case of the Phase–I CMS geometry, effects of multiple scattering and energy loss are accounted for by defining two-dimensional arrays for radiation and interaction lengths, indexed in $r - z$. These constants are taken from the CMS simulation. The MKFIT program supports the usage of both constant and parametrized magnetic fields.

3.2. Event handling and processing
Hit and seed data are inputs to the software. In the standalone operation mode, the data are read from a binary file created by a dedicated converter application. The same binary file can contain vectors of simulated tracks, and of tracks reconstructed by the standard CMS tracking, used to validate the physics performance of MKFIT. When operating within CMSSW, MKFIT is used as an external library, with a dedicated CMSSW processing module that is run within the CMSSW framework. This module is responsible for packaging the input hit and seed data in the format expected by MKFIT, and for providing the configuration for the MKFIT execution. After the execution, the tracks found by the MKFIT algorithm are copied from the MKFIT format into the CMSSW format.

Prior to the track finding, the seed collection undergoes a dedicated *seed cleaning* algorithm, aimed at removing multiple instances of seeds that likely originate from hits belonging to the same outgoing particle. This algorithm exploits the identity of hits, as well as the p_{T}, η, and ϕ seed parameters, and is tuned to preserve track finding efficiency in high PU events. Duplicate seeds arise from detector module overlaps, which are especially significant in the endcaps of the CMS tracker. Duplicate seeds are not removed prior to the standard CMS track finding, in CMSSW: CMSSW processes seeds one by one, and when a track candidate is found its hits are marked as used; if all the hits of a seed are marked as used from a previously found track candidate, that seed is not processed. This approach is not practical as it would be a serial bottleneck in MKFIT, where up to $32 \times N_{\mathrm{threads}}$ seeds are processed in parallel, and seeds are grouped to be close in η and ϕ to maximize the memory cache reuse of hit data.

4. Physics performance
The physics performance of MKFIT is validated in simulated $t\bar{t}$ events with PU of 50 or 70, using the Phase–I CMS detector geometry. A constant magnetic field of 3.8 T is used. Results correspond to the CMS *initialStep* tracking iteration, where seeds are required to have 4 hits coming from distinct inner pixel layers, and to be compatible with the beam spot constraint [3]. Equivalent results from the standard CMS tracking are also shown for comparison, using the same set of input seeds.

The results shown in this section must be considered as preliminary, since work is ongoing, as of the time of the conference, to finalize and optimize the MKFIT algorithm. For instance:

- hit search windows, track candidate scoring, and final track quality criteria are still undergoing optimization;
- cleaning and merging of the final track collection is not yet implemented, which would include removal of duplicate tracks due to multiple seeds per simulated particle.

The physics performance of MKFIT is evaluated and validated using two validation suites:

- standalone validation: only used for the standalone MKFIT operation mode, aimed at evaluating the MKFIT algorithm-level efficiency;

- CMSSW validation: aimed at evaluating and validating the absolute performance of mkFit.

In the standalone validation: reference tracks are simulated tracks with hits on at least 12 layers of the CMS tracker, including 4 layers from a seed track; a reconstructed track with $N_{\text{hits}} \geq 10$, with 4 of these hits belonging to a seed track, is considered matched to a reference track if at least 50% of its hits belong to the latter, in addition to the 4 seed track hits. In the CMSSW validation (as shown in this document): reference tracks are simulated tracks with $p_{\text{T}} > 0.9$ GeV, $|\eta| < 2.5$, and $|d_{\text{xy}}(\text{vertex, beam axis})| < 3.5$ cm; a reconstructed track is considered matched to a reference track if $> 75\%$ of its hits belong to the latter, with no additional selection and no seed hit requirement. In both cases, track reconstruction efficiency is defined as the fraction of reference tracks with at least a matching reconstructed track. Similarly, the duplicate track rate is defined as the fraction of reference tracks with more than one matching reconstructed track.

Figure 1 shows the track reconstruction efficiency as obtained from the standalone validation suite, for both mkFit and CMSSW tracks, as a function of the track η, in simulated $t\bar{t}$ events with PU of 70: the mkFit algorithm-level efficiency is as good as the standard CMSSW track reconstruction, or larger. The duplicate track rate as obtained from the standalone validation suite in the same simulated events, for both mkFit and CMSSW tracks, is shown in Fig. 2 as a function of the track η: the mkFit duplicate rate is especially large for endcap tracks, mainly due to module overlaps in this region of the CMS tracker, and to the absence, in mkFit, of a dedicated duplicate track removal procedure. An effort is ongoing for the implementation of such a procedure after the track finding.

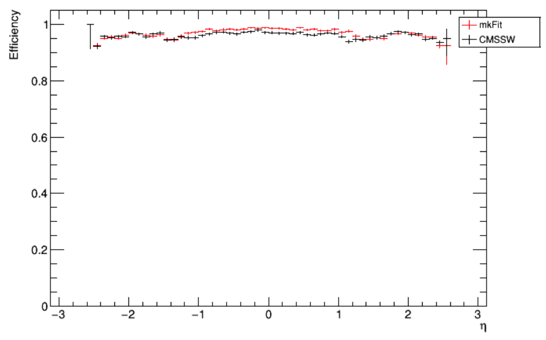

Figure 1. Track reconstruction efficiency, as obtained from the mkFit standalone validation suite, as a function of the track η for tracks with $p_{\text{T}} > 0.9$ GeV. The threshold $p_{\text{T}} > 0.9$ GeV corresponds to the target minimum p_{T} threshold for CMS HLT operation.

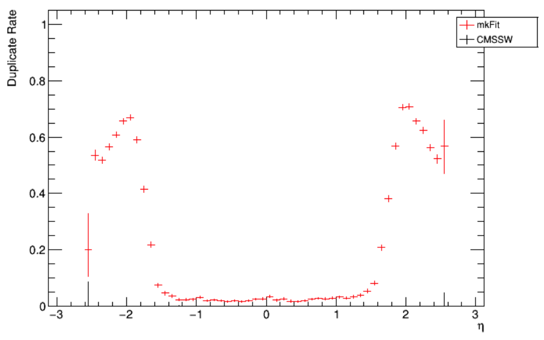

Figure 2. Duplicate track rate, as obtained from the mkFit standalone validation suite, as a function of the track η for tracks with $p_{\text{T}} > 0.9$ GeV. The threshold $p_{\text{T}} > 0.9$ GeV corresponds to the target minimum p_{T} threshold for CMS HLT operation. CMSSW values are nearly zero.

Figure 3 shows the track reconstruction efficiency as obtained from the CMSSW validation suite, for both mkFit and CMSSW tracks, as a function of the number of tracker layers, in simulated $t\bar{t}$ events with PU of 50, when mkFit is operated within CMSSW: a relative inefficiency of mkFit is observed, compared to CMSSW tracks, for tracks with $N_{\text{layers}} < 12$. The inefficiency is related to partial mkFit algorithm optimizations based on the mkFit standalone validation suite. Work is ongoing to recover the lost efficiency. The preliminary results of a test performed for this purpose are shown in Fig. 3.

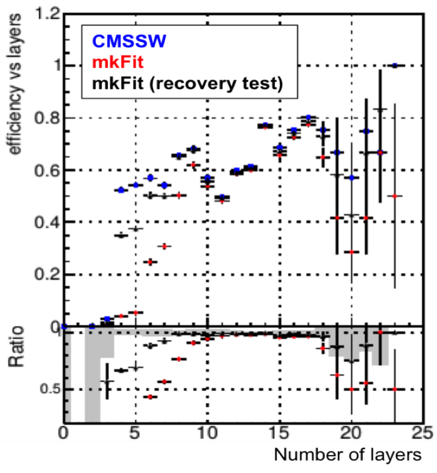

Figure 3. Track reconstruction efficiency in simulated $t\bar{t}$ events with PU of 50, as obtained from the CMSSW validation suite, as a function of N_{layers}, for CMSSW tracks, MKFIT tracks, and MKFIT tracks where a test was performed to (partially) recover the inefficiency otherwise observed for tracks with $N_{\text{layers}} < 12$.

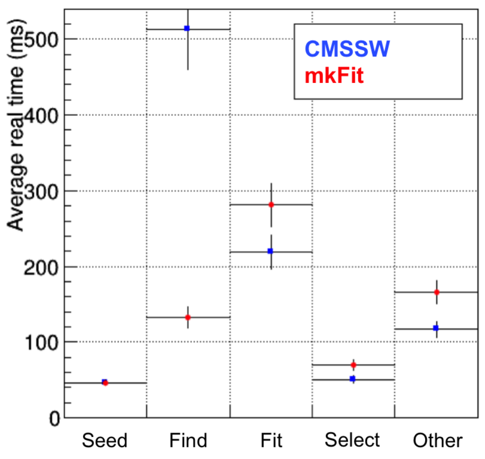

Figure 4. Average time per event for each step of the track reconstruction, for both MKFIT and CMSSW tracks, when MKFIT is operated within CMSSW, in simulated $t\bar{t}$ events with PU of 50, on SKL–SP, Skylake Gold, Intel Xeon Gold 6130 CPU @ 2.1 GHz, using a single thread. Each bin is a step of the track reconstruction.

5. Computational performance

The computational performance is evaluated on the most recent development platform: SKL–SP, Skylake Gold, 2 sockets × 16 cores with hyperthreading enabled, Intel Xeon Gold 6130 CPU @ 2.1 GHz, with the Turbo Boost feature disabled. The Intel `icc` compiler is used, together with the `AVX-512` set of instructions.

The track reconstruction performance is measured in simulated $t\bar{t}$ events with PU of 50, using a single thread. Figure 4 shows the average time per event for each step of the track reconstruction in the case of MKFIT tracks and standard CMSSW tracks.

For track finding, MKFIT is found to be faster than CMSSW, by a factor of about 4. This factor also accounts for the data format conversions described in Sec. 3.2, which make up about 40% of the total MKFIT track finding time. Once the data conversion operations have been optimized, MKFIT can be expected to achieve even better performance compared to CMSSW, by up to a factor of about 7, in the ideal case of no time spent in data format conversions. The performance advantage of MKFIT, as compared to CMSSW standard tracking, is understood as coming from two sources: vectorization, and different choices with respect to CMSSW for the implementation of geometry and detector description (see Sec. 3.1).

For both the MKFIT and CMSSW results, final track fitting is performed within CMSSW using its standard fitting procedure. In addition to the track KF fit, this procedure includes hit outlier rejection, based on a precise cluster position estimate for the pixel layers of the detector. Interestingly, the final track fitting takes longer for MKFIT. This can be attributed to the larger duplicate track rate observed for MKFIT. The in-progress MKFIT duplicate track removal procedure is expected to cancel the currently observed deficit of performance.

The MKFIT algorithm is also able to exploit multithreading, and when this is enabled within each event, the per-event timing is demonstrated to scale with the number of threads as shown in Fig. 5. In this case, track finding is performed on simulated $t\bar{t}$ events with PU of 70, using the standalone MKFIT operation mode, on the same SKL–SP architecture.

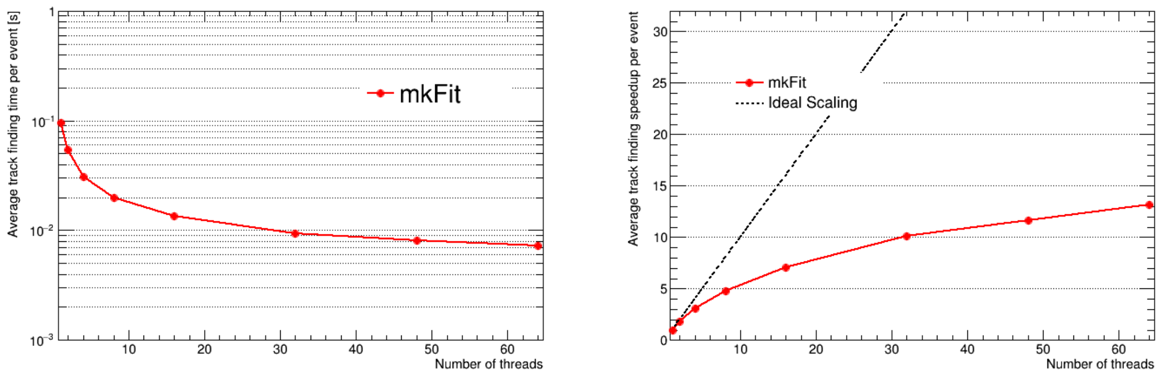

Figure 5. Average mkFit track finding time per event (left) and corresponding speedup (right) as a function of the used number of threads, on SKL–SP, in simulated $t\bar{t}$ events with PU of 70, using the standalone mkFit operation mode.

6. Conclusion and outlook

An algorithm for parallelized Kalman-Filter-based reconstruction of charged-particle tracks on multi-/many-core architectures has been developed, and its performance has been demonstrated using the CMS detector at the LHC, with conditions that reflect what is expected during the LHC Run III. The preliminary results summarized in this document show that mkFit can achieve comparable physics performance with respect to traditional KF tracking algorithms, with residual relative inefficiencies being recoverable by tuning the algorithm parameters, while retaining a significant improvement in computational performance. Ongoing work focuses on the optimization of the track finding algorithm parameters, as well as on the implementation of the final track post-processing steps.

Integration of mkFit with CMSSW is also advancing, on a path for a full integration in the CMS HLT test-bed system for the LHC Run III. In this context, work is ongoing on the optimization of data format conversion operations. Future work will focus on the implementation of the Phase–II CMS detector, with the ultimate goal of employing mkFit in the CMS HLT, and possibly offline, during the HL–LHC era.

Acknowledgments

This work is supported by the U.S. National Science Foundation, under grants PHY–1520969, PHY–1521042, PHY–1520942 and PHY–1624356, and under Cooperative Agreement OAC-1836650, and by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program. The authors wish to recognize the intellectual environment provided by the CMS collaboration.

References

[1] The CMS Collaboration, *JINST* **3** (2008) S08004
[2] R. Frühwirth, *Nucl. Instrum. Meth.* **A262** (1987) 444-450
[3] The CMS Collaboration, *JINST* **9** (2014) no.10, P10009
[4] G. Cerati *et al.*, *J. Phys. Conf. Ser.* **608** (2015) no.1, 012057
[5] G. Cerati *et al.*, *J. Phys. Conf. Ser.* **664** (2015) no.7, 072008
[6] G. Cerati *et al.*, *EPJ Web of Conferences* **127** (2016) 00010
[7] G. Cerati *et al.*, *J. Phys. Conf. Ser.* **898** (2017) no.4, 042051
[8] G. Cerati *et al.*, *J. Phys. Conf. Ser.* **1085** (2018) no.4, 042016
[9] G. Cerati *et al.*, *EPJ Web of Conferences* **150** (2017) 00006