Particle tracking in CMS with mkFit: from Run3 to HL-LHC

Steven Lantz,[a] Michael Reid,[a] Daniel Riley,[a] Peter Wittich,[a] Giuseppe Cerati,[b*] Matti Kortelainen,[b] Allison Reinsvold Hall,[b] Peter Elmer,[c] Bei Wang,[c] Leonardo Giannini,[d] Vyacheslav Krutelyov,[d] Mario Masciovecchio,[d] Matevž Tadel,[d] Frank Würthwein,[d] Avraham Yagil,[d] Brian Gravelle,[e] Boyana Norris.[e]

[a]Cornell University, Ithaca, NY, USA 14853
[b]Fermi National Accelerator Laboratory, Batavia, IL, USA 60510
[c]Princeton University, Princeton, NJ, USA 08544
[d]UC San Diego, La Jolla, CA, USA 92093
[e]University of Oregon, Eugene, OR, USA 97403

(*) email contact: cerati_AT_fnal.gov

Introduction

The reconstruction of the trajectory of charged particles is the most time consuming task in the data processing of LHC [1] experiments. It is a combinatorial algorithm, so its complexity scales worse than linearly with increasing detector occupancy. With higher beam luminosity, the detector occupancy increases mainly because of the larger number of simultaneous proton collisions per bunch crossing, also referred to as pile-up (PU). A dramatic increase in luminosity, and thus of PU, is expected at the High-Luminosity LHC (HL-LHC) [2], currently foreseen in 2027. With expected PU as high as 200 the processing time will soar, so a major update to the tracking algorithms needs to be completed before the HL-LHC.

A shortcut to reduce the processing time can be to reduce the phase space of the reconstructed tracks, limiting the reconstruction to high momentum tracks attached to the primary interaction vertex. However, this approach compromises the physics reach of the experiments, reducing the sensitivity to displaced signatures or impeding the application of PU-suppression techniques. A different solution is offered by the recent developments in the computing architecture domain, with the emergence of many-core CPUs and GPUs. These architectures feature data- and instruction-level parallelism that, if exploited, can lead to large speedups.

This letter reports the status and plans of mkFit [3], a project whose goal is to deploy a Kalman filter-based [4,5] parallel tracking algorithm in CMS [6].

Summary of mkFit Algorithm

mkFit is a track reconstruction algorithm that leverages data- and instruction-level parallelism, leading to large speedups compared to the state-of-the-art algorithm. Full details can be found in the recent mkFit paper [3], so only a summary is reported here.

The mkFit algorithm is designed for parallelization from the start. In order to exploit data parallelization we developed Matriplex, a library for optimal SIMD implementation of small matrix operations. The main concept is that matrices are organized in batches, with size matching the one of vector registers, and all matrices within a batch operate in lock-step. Task-level parallelism is implemented by using TBB at three different levels: a coarse parallelism at event level, an intermediate in different detector regions, and a fine-grained one at track candidate level. In order to minimize memory-bandwidth bottlenecks, data products are kept to a minimum size, and detector information, such as geometry and material, are described with a simplified and lightweight approach.

Results show that the physics performance, when evaluated in terms of efficiency, fake rate, and number of found hits, is comparable with the state-of-the-art tracking algorithm used in CMS. Scaling tests on a standalone application of the mkFit algorithm demonstrate that significant speedups are obtained thanks to parallelization: vectorization leads to up to 3x faster execution speed and multithreading scales well giving speedups exceeding the number of available physical cores when utilizing hyperthreading. The mkFit algorithm is integrated in the CMS data processing software (CMSSW) [7], and, when executed in a single-thread job, is found to be about 6x faster than the nominal algorithm; it is worth stressing that in order to achieve this result, a key is to compile mkFit with the icc compiler and enabling the AVX2 or AVX512 instruction sets.

Ongoing activities and prospects

After demonstrating the potential for large speedups with comparable physics performance, the main focus of the mkFit project is the deployment in the experiment's reconstruction workflows. This is a delicate step, as the tuning of the algorithm's parameters as well as the interplay with the rest of the reconstruction chain can make a difference in terms of overall physics performance. The goal is to define mkFit-based configurations for both the CMS offline and online (High-Level Trigger, or HLT [8]) tracking in time for the start of the LHC Run3. Challenges for the two configurations are different. In the offline configuration, the balance between computing speed and physics accuracy leans towards the latter. The CMS offline tracking consists of a series of iterations, aiming at reconstructing tracks that are of increasing difficulty. While its approach is generic, mkFit was developed and tested using the first iteration as a reference. Work is ongoing to add support for multiple iterations in mkFit, and to tune its performance for different iterations. In the online configuration, instead, timing constraints impose stricter requirements on the processing speed. This is complicated by the fact that what matters is the speed of a full trigger path, so that input and output algorithms to tracking need to be taken into account as well. In addition, for many paths, tracking is run only in a narrow region of the phase space, so the overall workload is reduced and the benefits from parallelism are reduced. Thus the challenge is to restructure the iterations in a way that can fully exploit the speedup potential brought in by mkFit.

In parallel, we are planning to enable the execution of mkFit with the CMS Phase2 tracker detector geometry, which will be used during the HL-LHC. The development plan includes different stages that do not necessarily need to be completed sequentially. First, the code needs to be adapted to the different geometry layout; this should be a relatively simple task given that geometry information is used in a lightweight fashion, factorized from the core functions. Then, we will explore how new features of the Phase2 CMS detector [9] can be exploited in mkFit; these include the utilization of "vector hits" (i.e. hits carrying not only position but also momentum information) and possibly the utilization of hits from timing layers [10]. Finally, the interplay with more naturally parallelizable algorithms (not based on Kalman filtering), such as the segment linking [11], will be explored.

Another set of ongoing activities focus on the utilization of GPUs for tracking. In the context of the HLT deployment work, we are developing a GPU-based Silicon Strip detector reconstruction chain [12] for Run3. The chain includes several steps: the raw data unpacking, the clustering, and the hit position estimation. Strip hits are the main input to the mkFit algorithm, so speeding up their production can be a significant improvement to the timing of mkFit-based HLT paths. Also, we are exploring the utilization of dedicated libraries and compiler directives for portable implementations of the code across different architectures, such as CPUs and GPUs; the code used to benchmark the performance of the different solutions is based on a simple function from the mkFit repository so the results of these exploration may inform future developments, although scaling the tests up to the full mkFit code is currently not in the plans.

Conclusions

The mkFit project pioneered the parallelization of the Kalman-filter based track reconstruction algorithm used in HEP experiments. It successfully demonstrated that by re-designing the algorithm so that it exploits both data- and thread-level parallelism significant speedups can be obtained without compromises in terms of physics performance. Work continues towards the deployment in the CMS reconstruction sequences, with the end goal of providing a sustainable tracking solution for HL-LHC. In broader terms, lessons learned and specific solutions can inform efforts in the broader community.

References

[1] L. Evans and P. Bryant, LHC Machine, JINST 3 (2008) S08001.

[2] G. Apollinari, I. Béjar Alonso, O. Brüning, P. Fessia, M. Lamont, L. Rossi et al., High-Luminosity Large Hadron Collider (HL-LHC), CERN Yellow Rep. Monogr. 4 (2017) 1.

[3] S.Lantz, et al, Speeding up Particle Track Reconstruction using a Parallel Kalman Filter Algorithm, arXiv:2006.00071, accepted for publication by JINST, 2020.

[4] R. Fruhwirth, Application of Kalman filtering to track and vertex fitting, Nucl. Instrum. Meth. A262 (1987) 444.

[5] CMS collaboration, Description and performance of track and primary-vertex reconstruction with the CMS tracker, JINST 9 (2014) P10009 [1405.6569].

[6] CMS collaboration, The CMS Experiment at the CERN LHC, JINST 3 (2008) S08004

[7] C. D. Jones, M. Paterno, J. Kowalkowski, L. Sexton-Kennedy and W. Tanenbaum, The new CMS event data model and framework, in Proceedings of International Conference on Computing in High Energy and Nuclear Physics (CHEP06), 2006, https://indico.cern.ch/event/408139/.

[8] CMS collaboration, The CMS trigger system, JINST 12 (2017) P01020 [1609.02366].

[9] CMS collaboration, The Phase-2 Upgrade of the CMS Tracker, Tech. Rep. CERN-LHCC-2017-009, CMS-TDR-014, 2017.

[10] CMS collaboration, Technical Proposal for a MIP timing detector in the CMS experiment Phase 2 Upgrade, Tech. Rep. CERN-LHCC-2017-027, LHCC-P-009, 2017.

[11] P. Chang, et al., Parallelizable Track Pattern Recognition in High-Luminosity LHC, Proceedings of CTD 2020, PROC-CTD2020-17 (2020).

[12] G. Cerati, et al., Parallelizing the Unpacking and Clustering of Detector Data for Reconstruction of Charged Particle Tracks on Multi-core CPUs and Many-core GPUs, Proceedings of CTD 2020, PROC-CTD2020-37 (2020).