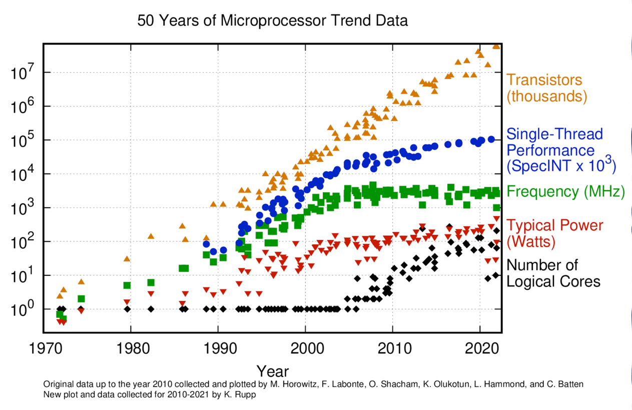
Speeding up the CMS track reconstruction with a parallelized and vectorized Kalman-filter-based algorithm during the LHC Run 3

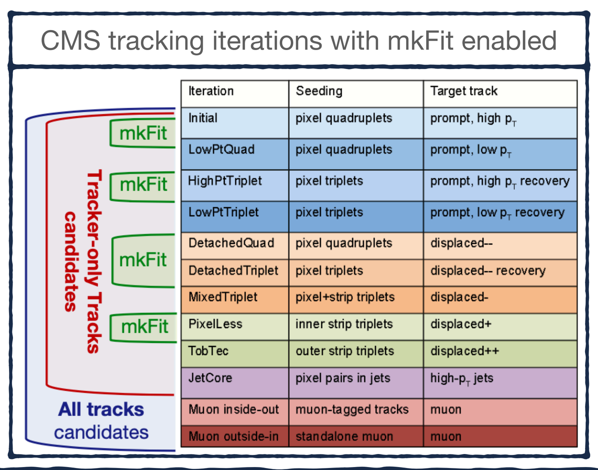
Emmanouil (Manos) Vourliotis[1] et al.* on behalf of the CMS Collaboration

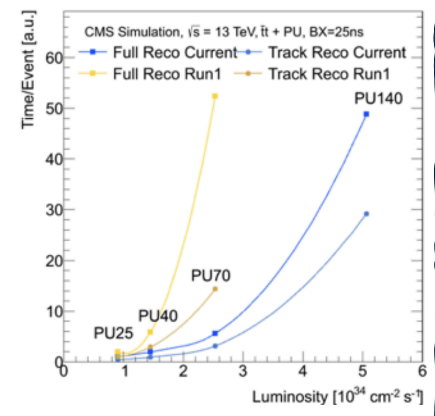
Motivation for Track Reconstruction Speed-up

- Exponential growth of CPU time/event at the LHC ⇒ Track reconstruction ~1/2 of event reconstruction.
- \+ Plateau of single thread performance.
- ⇒ Solution: **Parallelized & vectorized** tracking algorithms.



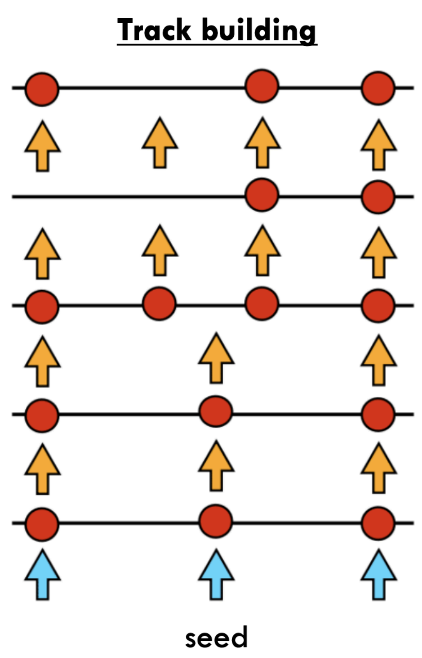
- **mkFit** Matriplex Kalman Fitter tracking algorithm[1]: Application of this paradigm shift → Goals:
  - **Same physics** performance.
  - Significant **speed-up**.
- mkFit in development for 5+ years → Now integrated in central CMS offline track building.





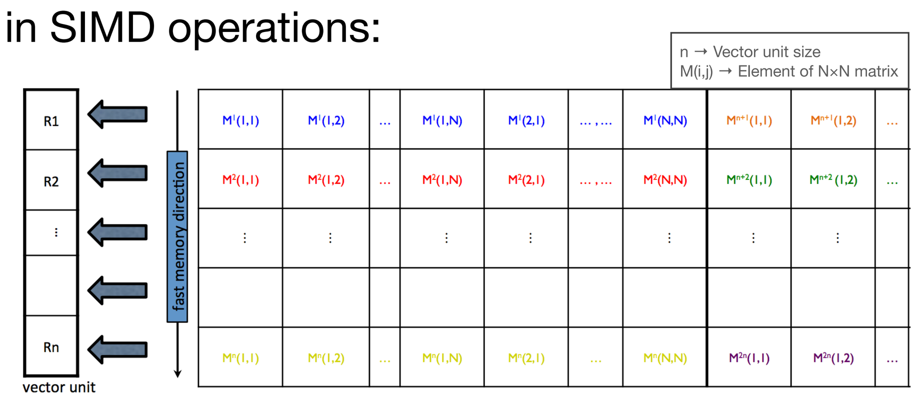
Algorithm Logic and Challenges

- CMS tracking[2] → Combinatorial Kalman Filter:
  - Start from track seeds.
  - Iteratively accumulate compatible hits to build tracks.
  - ⇒ The most time consuming part of track reconstruction.
- Improve by:
  - **Vectorization**: Same Instruction, Multiple Data (SIMD).
  - **Mutlithreading**: Different Instructions, Different Data.
- Requirements:
  - **Minimization of algorithm branching** for vector operations.
  - **Load balancing** among threads.
  - **Minimization** of memory usage and **optimisation of memory** accesses.



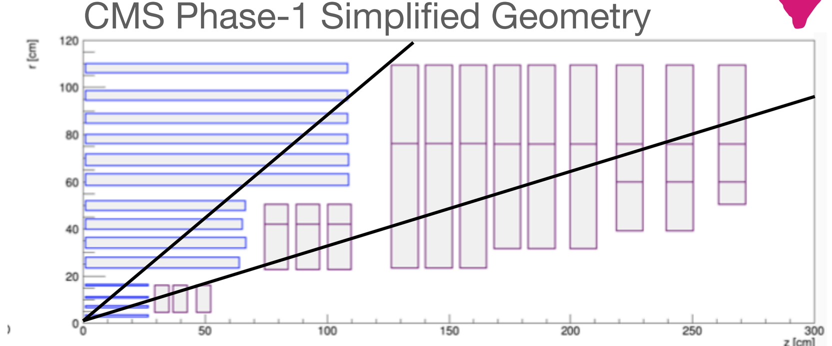
Parallelisation Strategy

- Multiple levels of multithread parallelization:
  - ↳ Loop over different events.
    - ↳ Loop over different η-regions.
      - ↳ Loop over z-/r- and ϕ-sorted groups of seeds.
- Exploit **Intel TBB** for task scheduling ⇒ Dynamic task stealing to balance workloads.
- Vectorisation of track candidate processing → Custom matrix library, **MATRIPLEX** ⇒ Optimized for vector loading in SIMD operations:
  - **"Matrix-major"** representation designed to fill a vector unit with **n** matrices in sync.



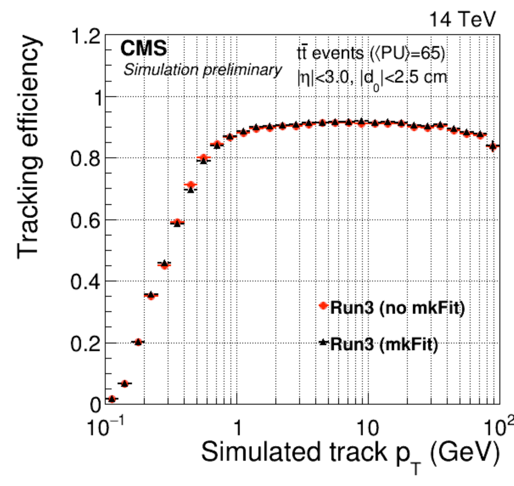
Simplified Detector Description

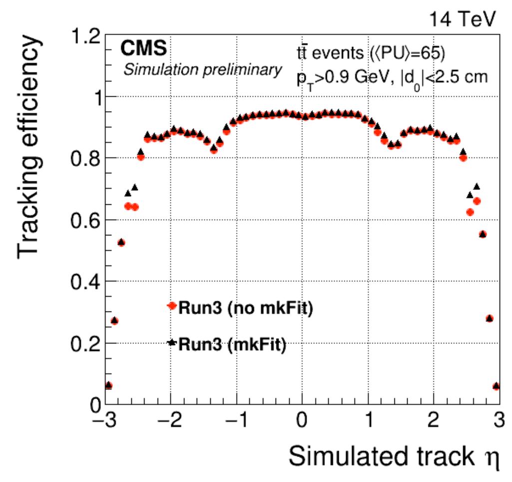
- Use highly configurable **tracker layer** data structures ⇒ Detector details in 2D (r/z, phi) bins instead of individual modules:
  - **Reduction of memory** usage.
  - Instruction **overhead minimization**.
- 2-step propagation:
  - Track → Average r or z ⇒ Create compatibility window.
  - Track → Each hit in window ⇒ Selection based on χ^2.
- **Multiple scattering** & **energy loss** effects included.

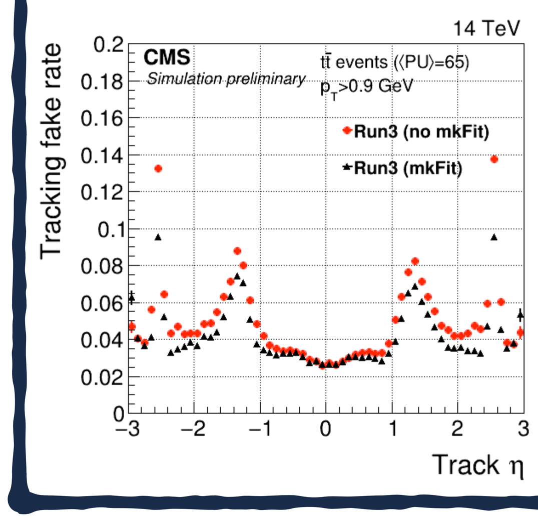


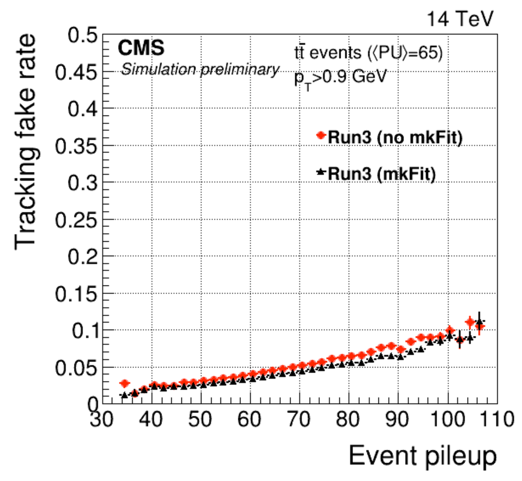
Physics Performance[3]

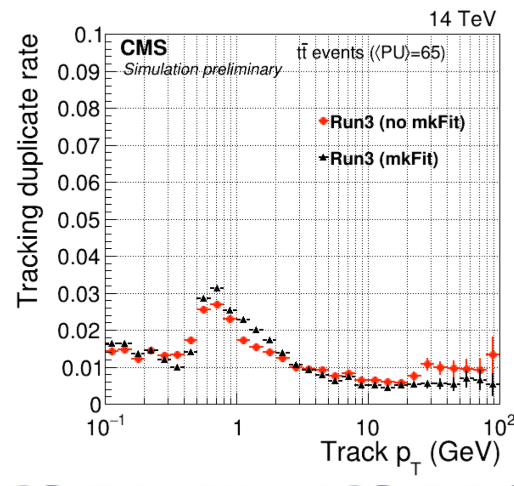
- mkFit used for almost 90% of all reconstructed signal tracks with $p_T > 0.5$ GeV.
- When using mkFit:
  - Tracking **efficiency comparable**:
    - Small gains in endcap ($2.4 < |\eta| < 2.8$).
  - Tracking **fake rate better** overall:
    - Fake rate reduction with increasing |η|.
    - Better scaling with PU.
  - Tracking **duplicate rate slightly increased** overall:
    - Due to parallel nature of the algorithm.
    - Mitigated with dedicated duplicate removal.
    - Performance tuned in p_T and η.





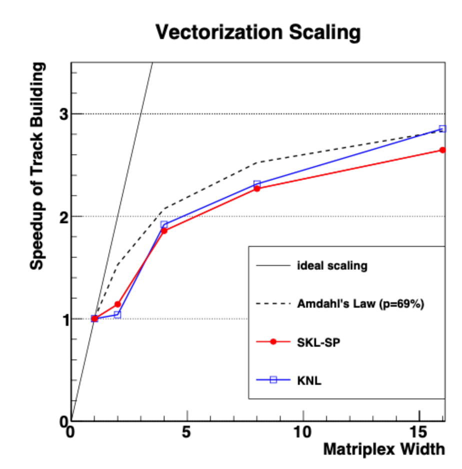


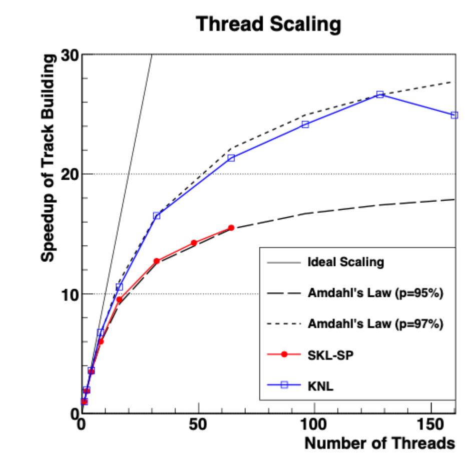


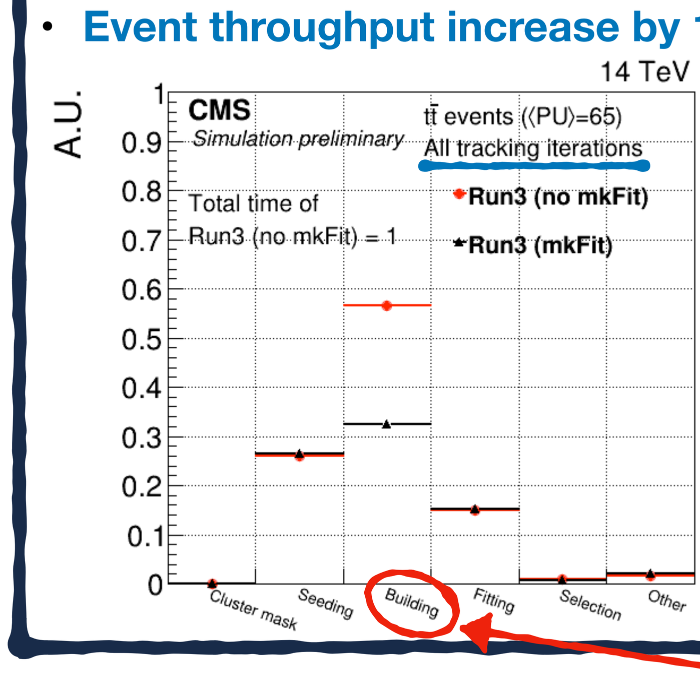


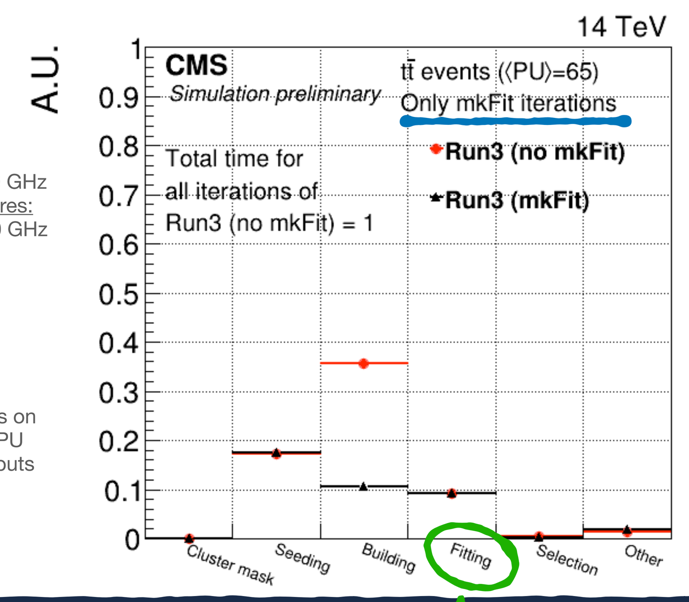
Timing Performance[3]

- mkFit vectorization & multithreading performance → According to Amdahl's Law:
  - **~70%** of operations effectively **vectorized**.
  - **>95%** of code effectively **parallelized**.
- When using mkFit:
  - **Individual mkFit** iterations: Up to **6.7x** building time reduction.
  - **Sum of mkFit** iterations: **~3.5x** building time reduction.
  - **Sum of all** iterations: **~1.7x** building time reduction ⇒ **25% reduction of total tracking time.**
- **Event throughput increase by 10–15%** in Run3.



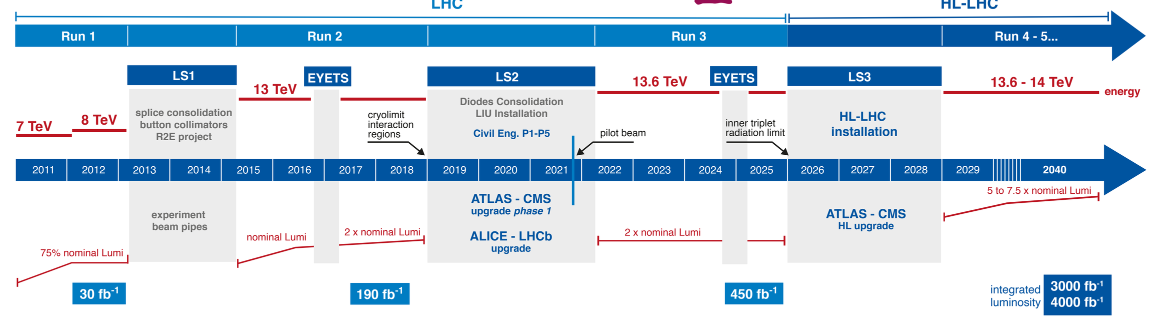






Outlook

- Plans for mkFit paradigm extension to more applications:
  - Extend to **more track building iterations** ⇒ Further speed-up.
  - Apply to **track fitting** → Its timing now comparable to track building.
  - Build tracks for **High Level Trigger**, already during Run3 → Global SiStrip RAW data unpacking needed, can now be on GPU.
  - Modify for **Phase-2 geometry and configuration**
    - Optimization and tuning.
    - Synergies with other algorithms.



*A. R. Hall[2], A. Yagil[1], B. Wang, B. Norris, B. Gravelle[5], D. S. Riley[6], G. Cerati[3], K. McDermott, L. Giannini[1], M. Masciovecchio[1], M. Tadel[1], M. Kortelainen[3], M. Gartung[3], P. Elmer[4], P. Wittich[6], S. Krutelyov[1], S. Berkman, S. R. Lantz[6], T. Reid[6]

[1]UCSD, [2]USNA Annapolis, [3]Fermilab, [4]Princeton, [5]Oregon, [6]Cornell

References:

[1] S. Lantz et al., "Speeding up particle track reconstruction using a parallel Kalman filter algorithm", arXiv:2006.00771, DOI:10.1088/1748-0221/15/09/P09030, JINST 15 (2020) 09, P09030

[2] CMS Collaboration, "Description and performance of track and primary-vertex reconstruction with the CMS tracker", arXiv:1405.6569, DOI:10.1088/1748-0221/9/10/P10009, JINST 9 (2014) 10, P10009

[3] CMS Collaboration, "Performance of Run 3 track reconstruction with the mkFit algorithm", CMS-DP-2022/018