Advanced tracking tools

Frank Gaede & Hadrien Grasland

DESY, Hamburg & LAL, Orsay

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 654168.

Task 3.6 – Advanced Tracking Tools

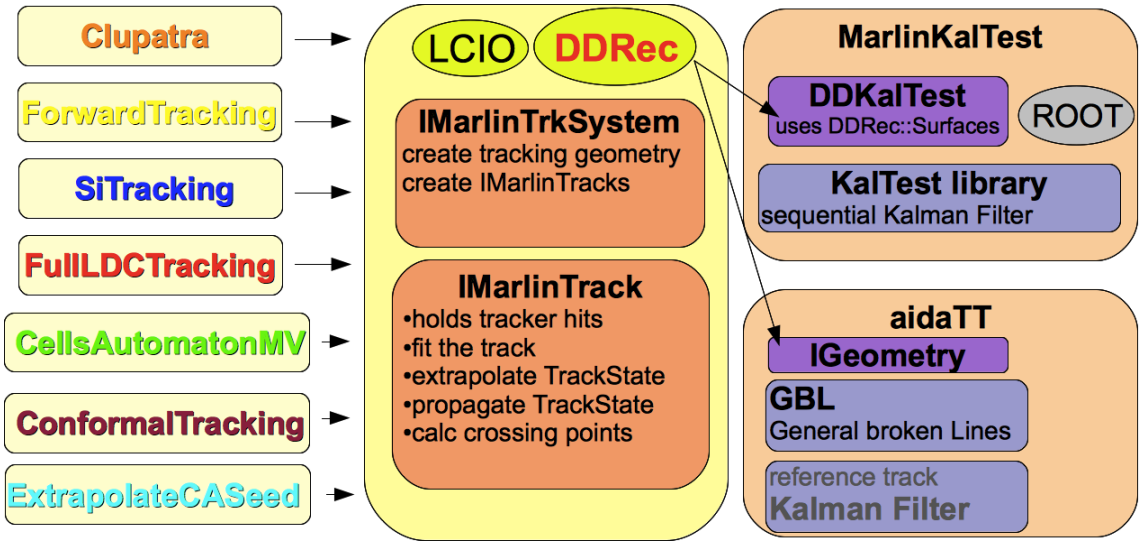
- Original task objectives (from AIDA2020 proposal):
  - Development of advanced parallel algorithms for track finding and fitting in AIDA Tracking Tool toolkit (**aidaTT**)
  - Application to LHC and LC
- Since then, ACTS was released as open source software
  - Based on ATLAS Run2 tracking software
  - Used for FCC, use planned for ATLAS Run3, interest from LC
- Decided to invest a large fraction of the work in ACTS:
  - Parallelization and optimization of ACTS tools
  - Integration of generic pattern recognition tools from aidaTT
  - Investigate application of ACTS to LC software

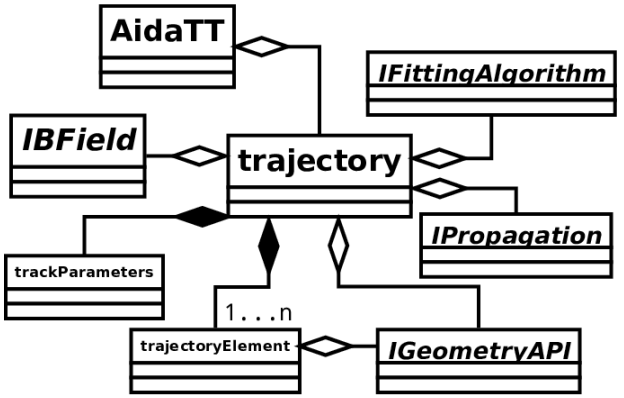
DESY Contribution

Introduction

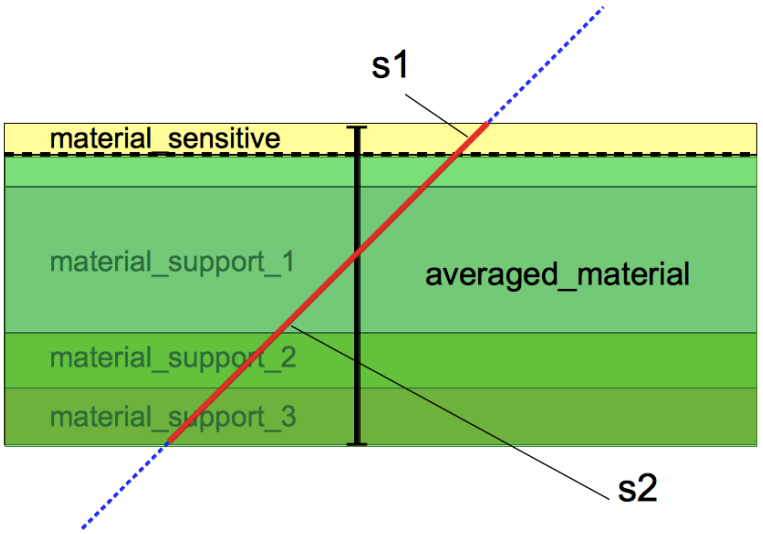
- Main focus at DESY for past year: finalizing the software chain for a large scale Monte Carlo production for ILD
- Based on software tools developed in AIDA2020:
  - DD4hep, DDRec, DDSim, aidaTT, MarlinTrk
- **Production has now started** (last week)
  - Has taken longer than anticipated, delayed planned work on further improving advanced tracking tools
- Some activities and plans related to advanced tracking tools are reported here

Tracking tools for Linear Colliders

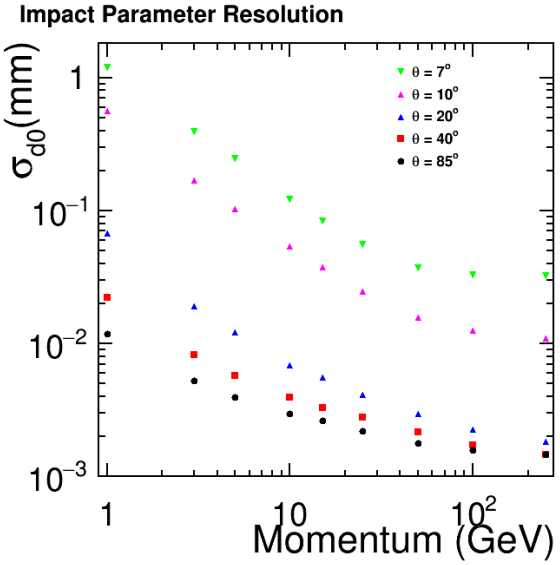


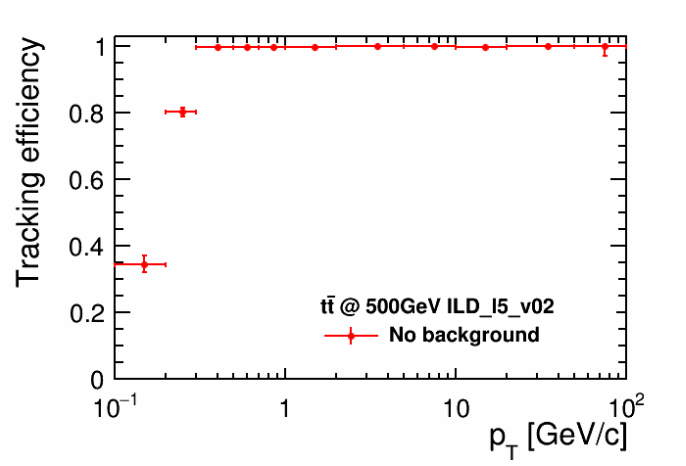


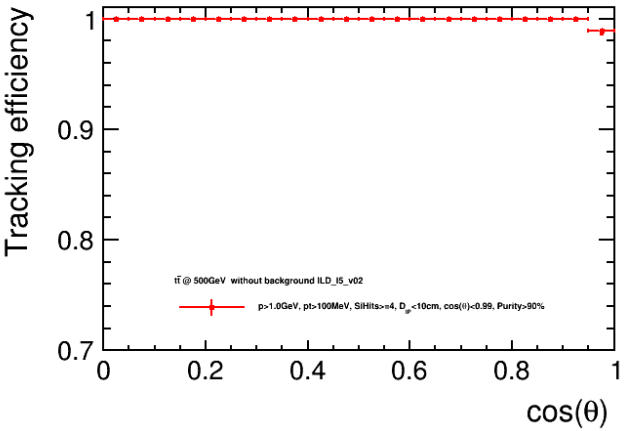
- Developed in AIDA/AIDA2020:
  - **MarlinTrk:** abstract interface to track fitting for ILCSoft framework (Marlin, LCIO, …)
  - **aidaTT:** interface for arbitrary track fitting tool
  - **DDRec:** geometry view for tracking in DD4hep-based detector models
    - uses *Surfaces* with material properties
  - Developed various – partly detector independent – **pattern recognition algorithms**
- Used by ILD, SiD, CLICdp (and FCC-ee)

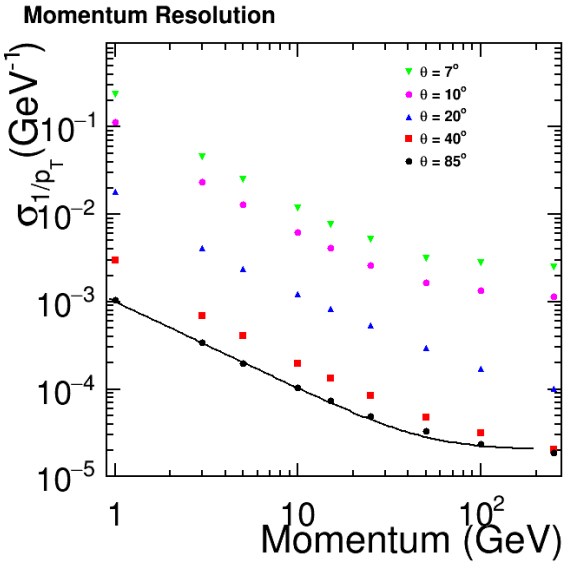


Tracking performance for ILD





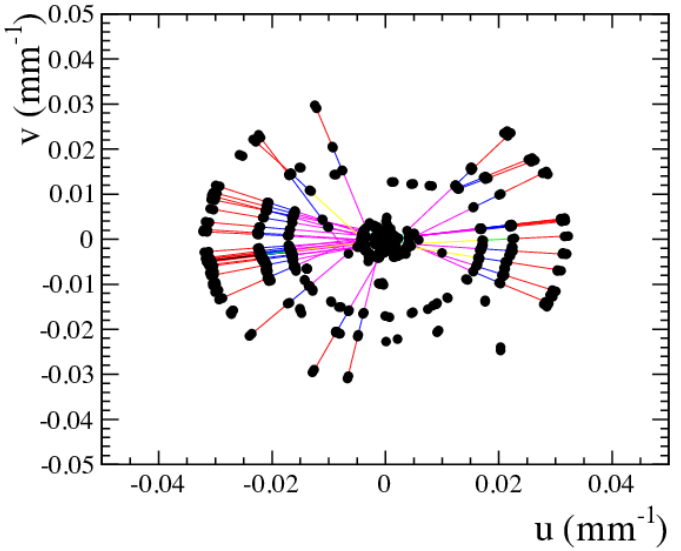




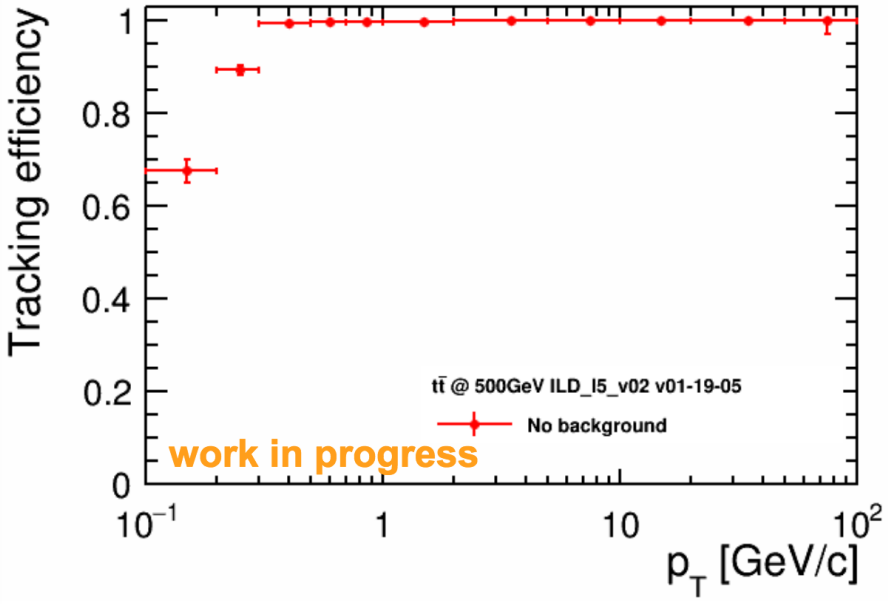
- Achieves excellent tracking performance
  - transverse momentum resolution
  - impact parameter resolution
  - tracking efficiency
    - perfect for pt > 300 MeV and cos(th) > 0.95
- Same as or better than old LC tracking SW

ConformalTracking for ILD

- pattern recognition developed for CLICdp
  - $u=x*r^{-2}$, $v=y^{-2}$, $r^2=x^2+y^2$
  - straight line search with *Cellular Automaton*
- code is rather **detector independent**



ConformalTracking



- Applied ConformalTracking to ILD
  - Si-trackers only (no TPC)
- Worked (almost) out-of-the-box
- Much improved efficiency at low p
- Could build a **truly detector- (and framework-) independent version**

Plans for remainder of project

- Move focus to improving the tracking tools
- Plan to investigate the possibility of using ACTS in the aidaTT/ MarlinTrk tracking interface and application to LC
  - Need to understand the best way to interface to ILCSoft
  - First prototype of tracking in CLICdp-like tracking detector with ACTS exists (CLICdp group)
- Plan to contribute some of the detector independent pattern recognition to ACTS examples
- Upcoming deliverable: **D3.7 Advanced Tracking tools M39**
  - Proposed to delay this into the extension period (if any)
  - Otherwise, must move it towards end of the regular project
  - Should know at the end of the annual meeting

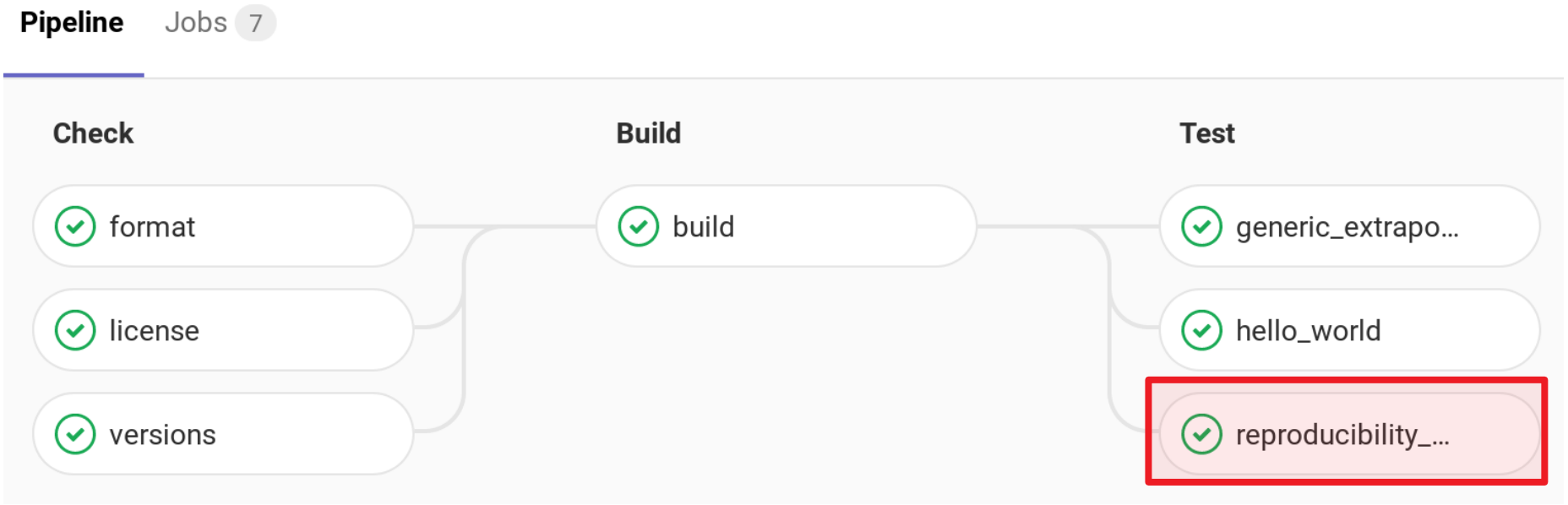
LAL Contribution

Previously…

- Decided to investigate ACTS, a promising tracking tool collaboration across experiment
- Studied obstacles to robust ACTS parallelization
  - Lack of multi-threaded validation
  - Const-incorrect interfaces
- Begun effort towards addressing these issues
  - Multi-threaded ACTS usage examples (+ scalability check)
  - Removed some const-incorrect code (mutable members…)
- Started R&D on improved usage of vectorization

Parallel ACTS validation

- ACTS' test framework now runs in parallel **by default**
- Parallel run results are **bitwise identical** to sequential case
- This is validated by the test framework's CI…



…and we already caught thread-safety bugs that way!

Const-correctness

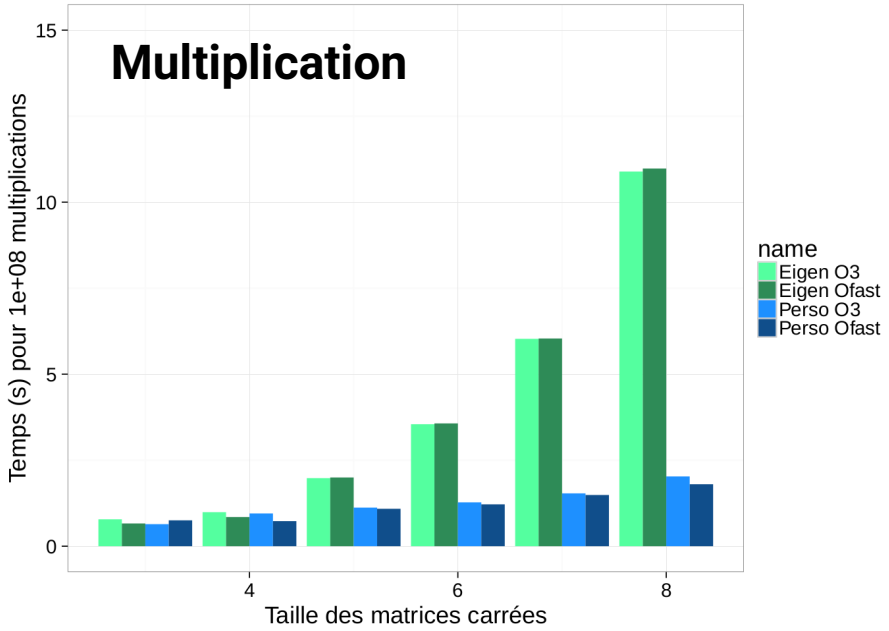
- Mutable class members are gone
- Static variables are gone
- Fixed a couple const-incorrect interfaces along the way
- Pointer-to-mutable is an unpleasant fellow
  - Frequently used for polymorphism
  - We want mutable during setup, const during processing
    - Sadly, there is no clean *and* cheap way to do this in C++
  - Current stopgap: use pointer-to-const + const_cast
  - Done for smart pointers, raw pointers are harder to review
    - Their syntax is overtly hostile to automatic search

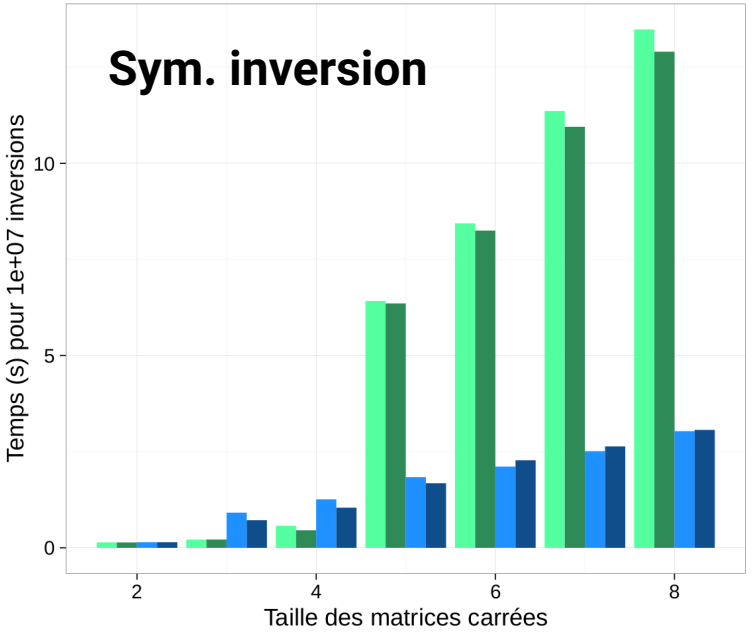
Extrapolation investigation

- ACTS provides two Runge-Kutta extrapolators
  - **Original ATLAS code:** Fast, well-validated, but unreadable
  - **Eigen-based rewrite:** Very clean… but measured 2.5x slower
- Nicolas Loizeau studied the sources of this discrepancy
  - Original comparison was unfair (different precision settings)
  - Basic optimizations missing (e.g. trigonometry avoidance)
  - Unnecessary coordinate conversions due to bad interface
- With help from A. Salzburger, they now perform identically
  - Maintainability or performance? Pick two!

Kalman Filter investigation

- Lucas Serrano investigated Kalman Filter vectorization
  - We previously knew that hand-written SIMD can outperform Eigen by factors of ~2x on 5x5 matrix algebra
  - Lucas proved that this can be done without sacrificing portability or a high-level code interface





Kalman Filter investigation (2)

- Sadly, Lucas' work could not be integrated in ACTS:
  - Expected PhD funding did not materialize, so he had to leave
  - Developers of the underlying SIMD toolkit (Boost.SIMD) pulled a tracherous move and broke the codebase
- However, we still get the benefit of insight:
  - We know that there is unexploited performance potential for small-matrix algebra, above the 1D-4D range of graphics[1]
  - We know it can be leveraged without losing maintainability

[1] …which is handled in Eigen using manually unrolled and hand-optimized code, with all the maintainability issues that this entails.

Prospect: xtensor investigation

- Lots of activity around the xtensor[3] stack in Orsay
  - Goal: Replicate numpy API feel & feature set in C++
- We would like to evaluate it as a possible Eigen successor
  - More familiar API for physicists with Python training
  - Larger developer base (Eigen is a ~1-man project)
  - Cleaner code (layered design: SIMD, N-d array, BLAS…)
  - …but can it handle small matrices, interoperate with Eigen?
- Taras Kolomatski will be joining us to perform this evaluation as a GSoC project

[3] http://quantstack.net/xtensor

Prospect: Numerical analysis

- We recently discovered Verrou[3], a Valgrind plugin for floating-point round-off error analysis
  - Performs **each** FP operation w/ random rounding mode
  - Allows studying global result fluctuations emerging from this localized last-bit noise
  - If a numerical instability is found, helps locate the cause
- I will evaluate applicability of this analysis to ACTS
  - As an automated complement to physics validation
  - As a helper in the study of reduced-precision arithmetic
  - Results will be presented at CHEP

[3] https://github.com/edf-hpc/verrou

Belle 2 activities

- LAL joined the Belle 2 collaboration last year
- I started investigating opportunities for common work:
  - Presented AIDA-2020 WP3 projects to them
  - ACTS and DD4Hep seem to have highest potential
  - Main concern: **Who will provide migration manpower?**
- On the tracking front specifically…
  - Desire to slowly replace Genfit: unmaintained, inefficient
  - Lack of manpower suggests piecewise ACTS integration
  - Question: Where does Belle 2 tracking spend its time?

Belle 2 tracking profile

- Profiling a Belle 2 tracking job revealed…
  - Extremely deep call graphs, suggesting inlining issues
  - Biggest identified bottleneck: Geant4 geometry lookups
  - Suspiciously high magnetic field lookup contribution
- These observations call for some actions/experiments:
  - Link-time optimization to enable more compiler auto-inlining
  - Manual inlining optimizations, if needed on the hot path
  - Update Geant4 to leverage VecGeom implementation
  - Review geometry layout, perhaps move to ACTS geometry?
  - Try out ACTS' recently optimized magnetic field map

Questions? Comments?