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 large fraction of the work in ACTS:
  - Parallelization of track finding and fitting tools
  - Integration of generic pattern recognition tools from aidaTT
  - Investigate application of ACTS to LC software

Tracking activities at DESY

- Validating AIDA tracking tools for large-scale MC production
- Checking automatic extraction of material description from DD4Hep to DDRec for tracking
  - Observing **compatible resolution** with former, manually defined tracking geometry
- Plan to investigate using ACTS in aidaTT/MarlinTrk tracking interface, for application to LC
  - ACTS has a tracking geometry prototype from CLICdp group
- H2 2017 will be more focused on tracking activities
- Remainder of this talk will focus on ACTS & LAL activities

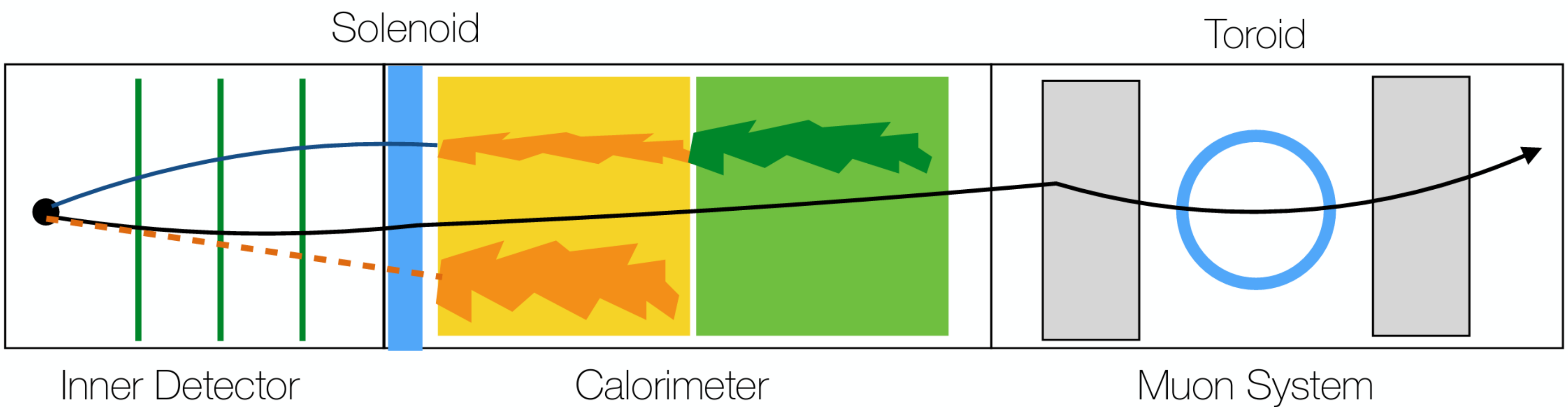
About ACTS

- “A Common Tracking Software”, http://acts.web.cern.ch/
- Extracting ATLAS’ tracking SW to an independent package
  - Benefits from decades of LHC bugfixes & optimization
  - Minimal build dependencies: CMake + Boost + Eigen

- Benefits of such a project:
  - Lets new experiments (FCC-hh, CLIC…) reuse previous work
  - Enables shared tracking R&D across experiments
  - Fosters collaboration with other fields (e.g. TrackML)
  - Streamlines use of standard software development tools

Why ATLAS?[1]

- ATLAS tracking is extremely complex:



- Hence their tracking code is built for diverse scenarios:
  - Two very different magnets → Field-agnostic code
  - Heterogeneous detection tech. → Technology-agnostic code
  - Big calorimeter in the middle → Integrated into tracking

[1] Slide contents from A. Salzburger, "ATLAS Tracking Software: history, status & prospects", Common Track Reconstruction Forum, https://indico.cern.ch/event/459865/contributions/1961745/attachments/1199090/1744232/CDOT-2015-Dec-Salzburger.pdf (2015)

Project status[2]

- What is available today:
  - Infrastructure (git workflow, CI + human review, Docker...)
  - Geometry (including DD4Hep & TGeo interfaces)
  - Event Data Model (highly configurable), incl. measurements
  - Extrapolation, propagation in magnetic field
  - Material mapping (from Geant4 to simplified geometry)

- What is being worked on:
  - Track finding and fitting
  - Documentation
  - Test coverage and thread safety

[2] See also: http://indico.cern.ch/event/577003/contributions/2476581/attachments/1424783/2185211/ACTS-CDOT-Status-2017-03-07.pdf

Thread-safety

- ACTS is not a reconstruction framework
  - It is a toolkit, to be integrated into experiment frameworks
  - Many of them moving to multi-threaded, multi-event designs
- ACTS comes remarkably well-prepared…

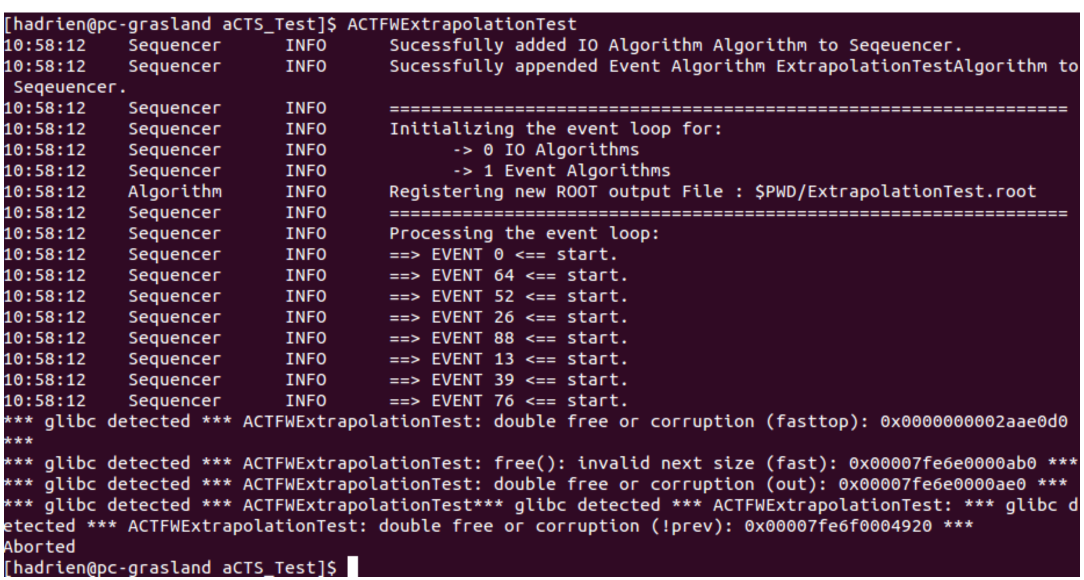
```
/** Templated RungeKutta propagation method - charged/neutral */
template <class T> bool propagateRungeKuttaT(ExtrapolationCell<T>& eCell,
                                             PropagationCache& pCache,
                                             const T& tParameters,
                                             const Surface& sf) const;
```

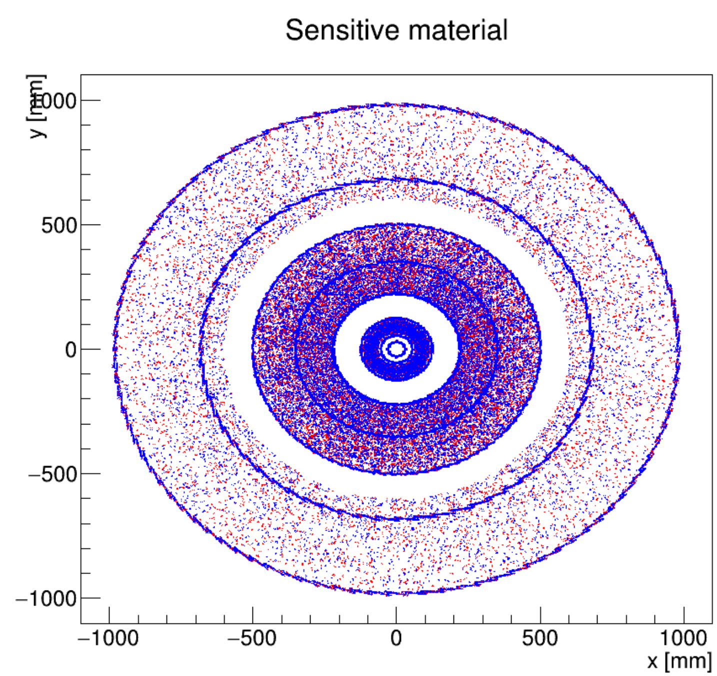
- ...but in C++, mistakes are only a few keystrokes away:

```
private:
  mutable std::vector<const DetectorElementBase*> m_binmembers;
  mutable std::vector<const DetectorElementBase*> m_neighbours;
};
```

Testing ACTS with threads

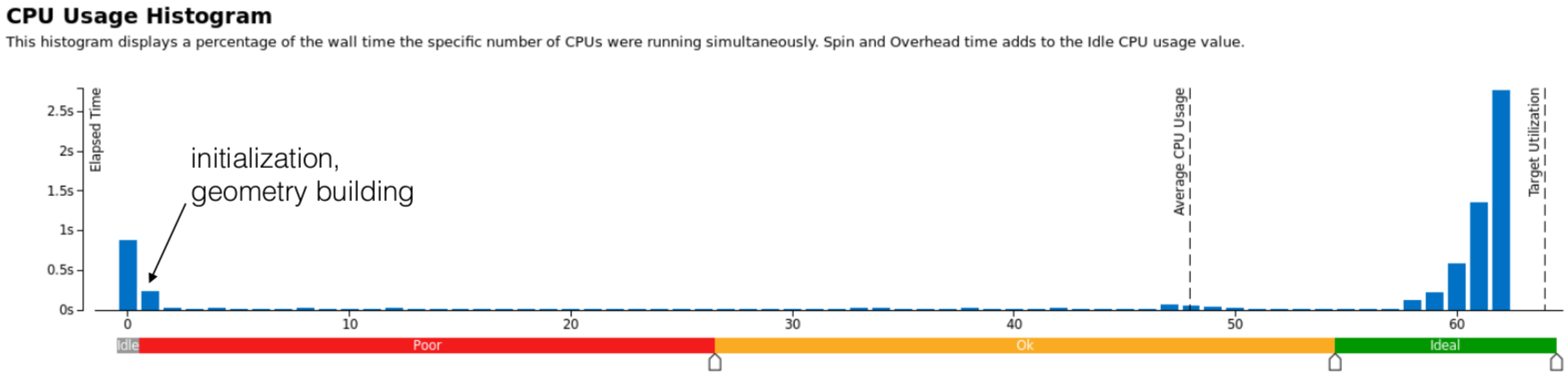
- First step: Made the test framework multithreaded
  - Testing new code with multiple threads should be trivial
  - Longer-term, I want to bring this to automated CI tests
- This uncovered many thread-safety issues… ☹
  - …in the test framework, not the toolkit itself ☺





Early performance numbers

- Extrapolation code scales well to highly parallel CPUs:
  - Tested on a dual-socket 64-core machine @ CERN openlab[2]
  - Workload: fast simulation without material effects
  - Essentially a stress test for propagation in magnetic field



Writing thread-safe code

- Top multi-threading benefit: Tasks can share data
  - Detector geometry, magnetic field, conditions…
- Top multi-threading hazard: Tasks are sharing data
  - Race conditions, complex synchronization, bottlenecks…
- Key to effective multi-threading: control data sharing
  - Avoid sharing mutable state (hard to get right, inefficient)
  - Try to restrict sharing to read-only data instead
  - In C++, a key ingredient is to enforce **const-correctness**

Steps towards const-correctness

- Ban careless use of “**mutable**” (const-incorrect, non-local)
  - Make interfaces const-correct whenever possible
  - When deeper refactoring is needed, temporarily replace with const_cast (still const-incorrect, but at least *local* to a scope)

Remove every use of "mutable" in ACTS MERGED 1 1 17
!265 · opened a month ago by Hadrien Grasland updated 3 weeks ago

- Review use of pointers in the ACTS codebase
  - C++ raw and smart pointers are const-incorrect by design ☹
  - Pointer-to-const (const T*) can sometimes be a workaround
  - Still looking for a good solution when mutation is needed…

Next steps

- DESY: Investigate adaptation of ACTS for ILC/CLIC
- LAL: Add automated multi-threaded tests to CI builds
- Review every single ACTS pointer for const-correctness
  - Use ptr-to-const more, improve ptr-to-mutable usability
- Optimize performance further through vectorization
  - Two interns coming at LAL, one aiming at a PhD
  - No shortage of subjects to tackle!
    - Parallel collision detection with VecGeom
    - Global-local coordinate transforms
    - Vectorized Runge-Kutta integration
    - Parallel Kalman filtering on CPUs and GPUs…

Questions? Comments?