Common Reconstruction Frameworks

G. Cerati (FNAL)

Software and Computing for Small HEP Experiments

Nov. 16, 2021

Common reconstruction tools: two examples

- LArSoft: simulation and reconstruction in LArTPC experiments
- ACTS: tracking for collider experiments

- I will give an overview about: concept, applicability, core functionalities, R&D

LArSoft

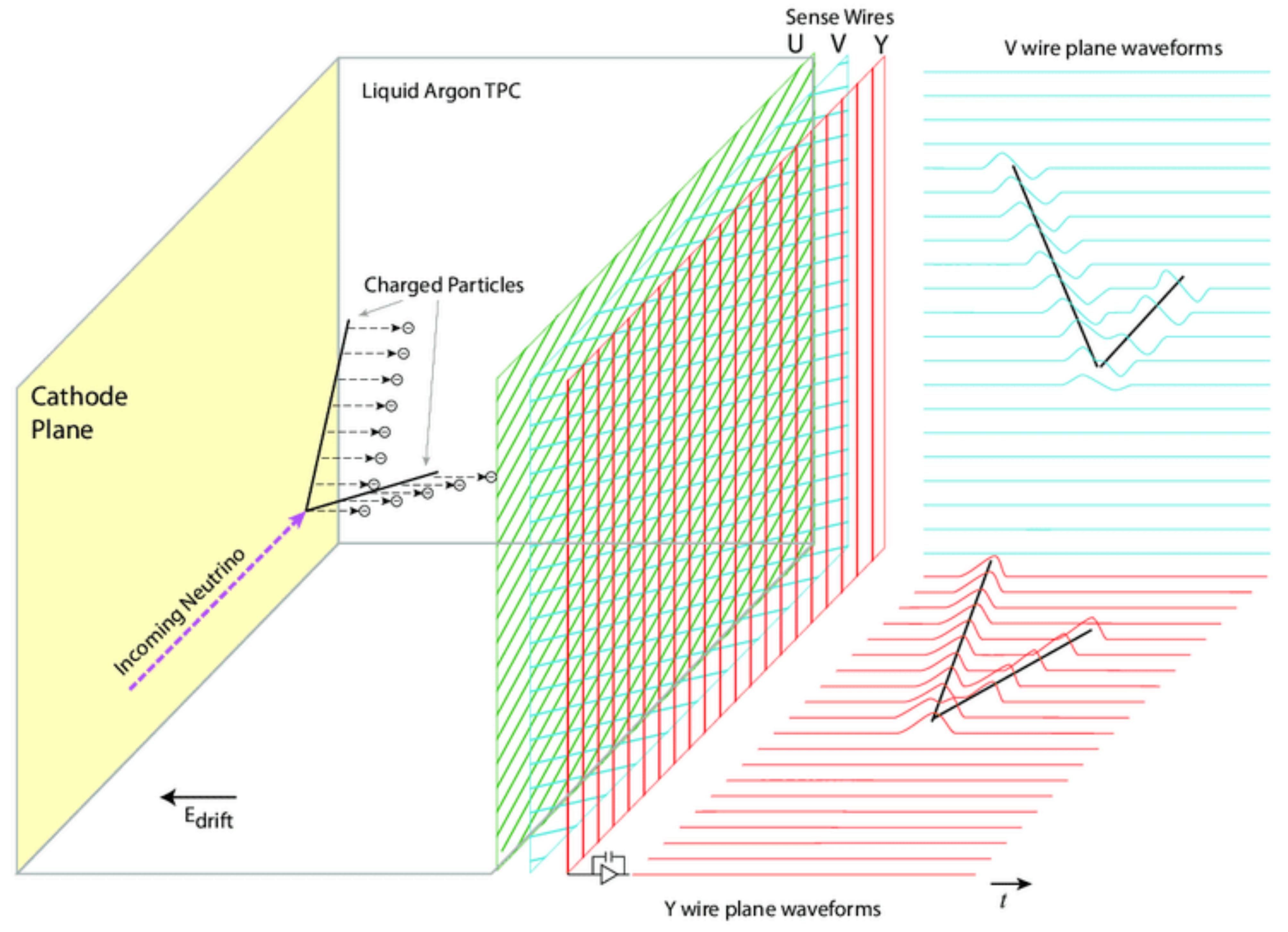
LArSoft

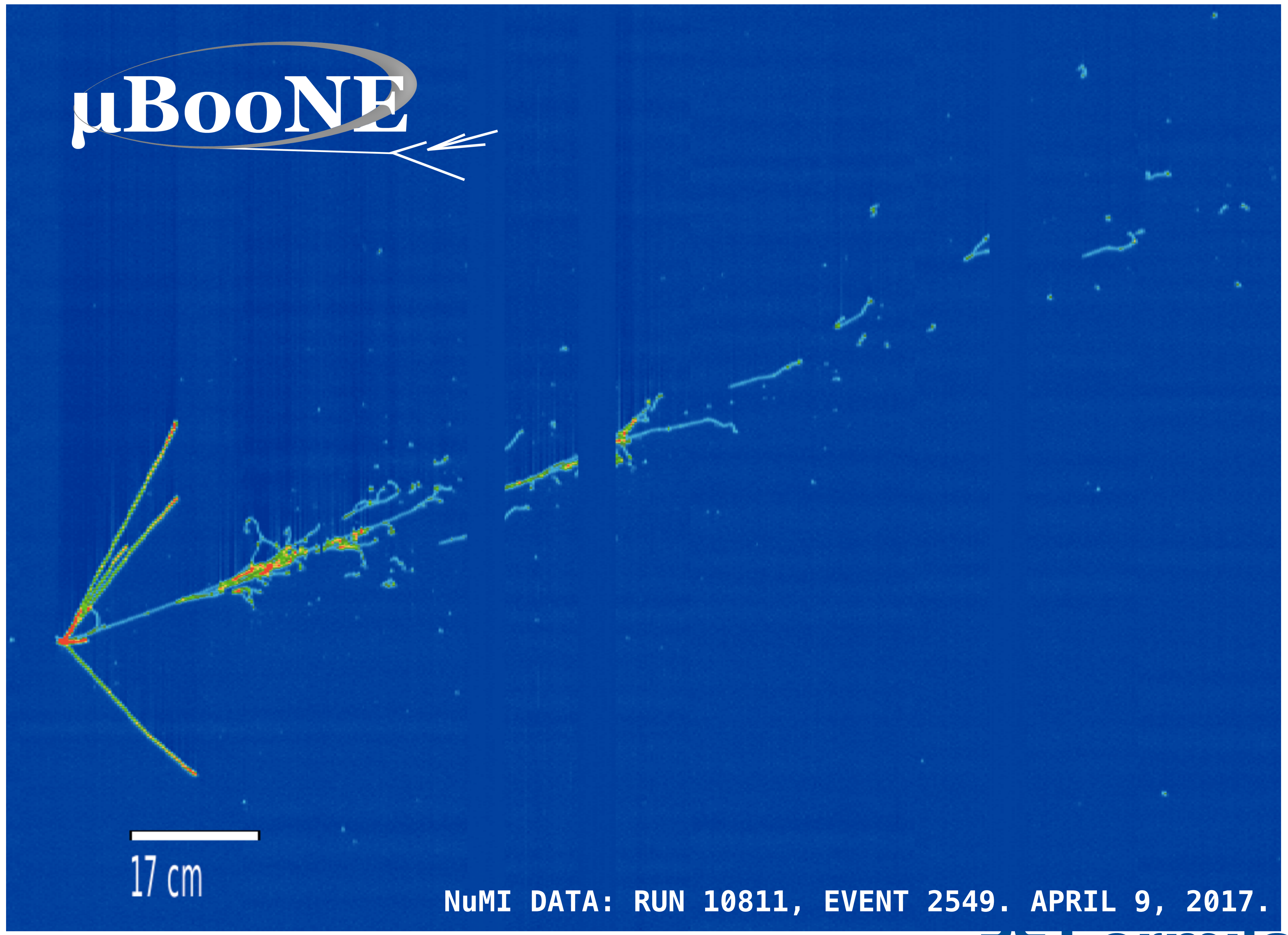
- Website: larsoft.org
- Paper: E.L. Snider and G. Petrillo 2017 J. Phys.: Conf. Ser. 898 042057
- Contact: scisoft-team@fnal.gov

- Most of the content stolen from Erica's ”Introduction to LArSoft”
  - https://indico.fnal.gov/event/49621/attachments/144683/184499/LArSoft%20introduction%202021-07-08.pdf

LArTPC Working Principle

- Charged particles produced in neutrino interactions ionize the argon, ionization electrons drift in electric field towards anode planes
- Sense wires detect the incoming charge, producing beautiful detector data images
- Fast scintillation light detected by PMT system (not shown below)





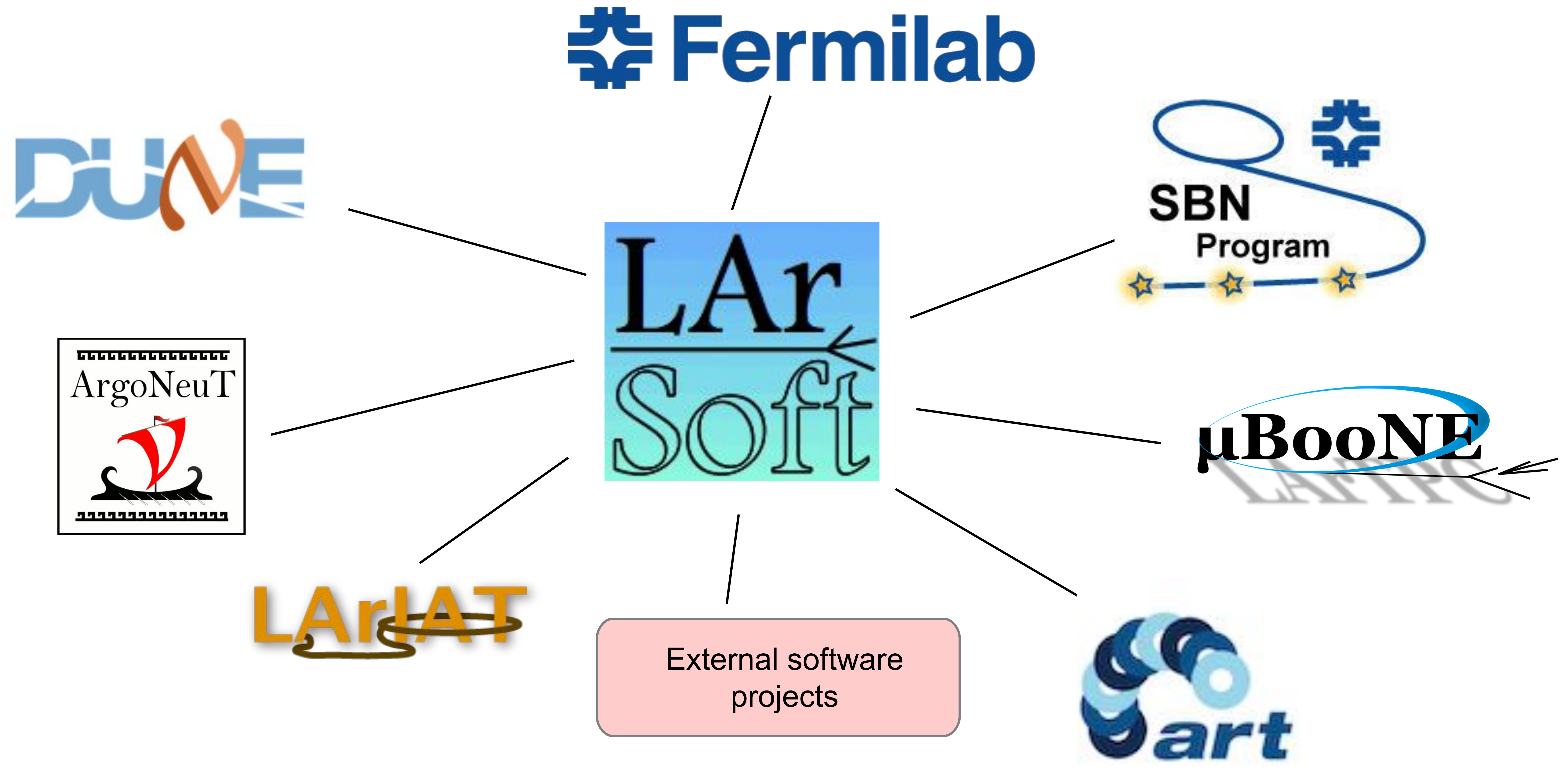
Underlying principle of LArSoft

Exploit the similarity in the geometry and readout schemes that are common to many LArTPCs to create a set of infrastructure and algorithms for the simulation and reconstruction of LArTPC data that can shared across detectors

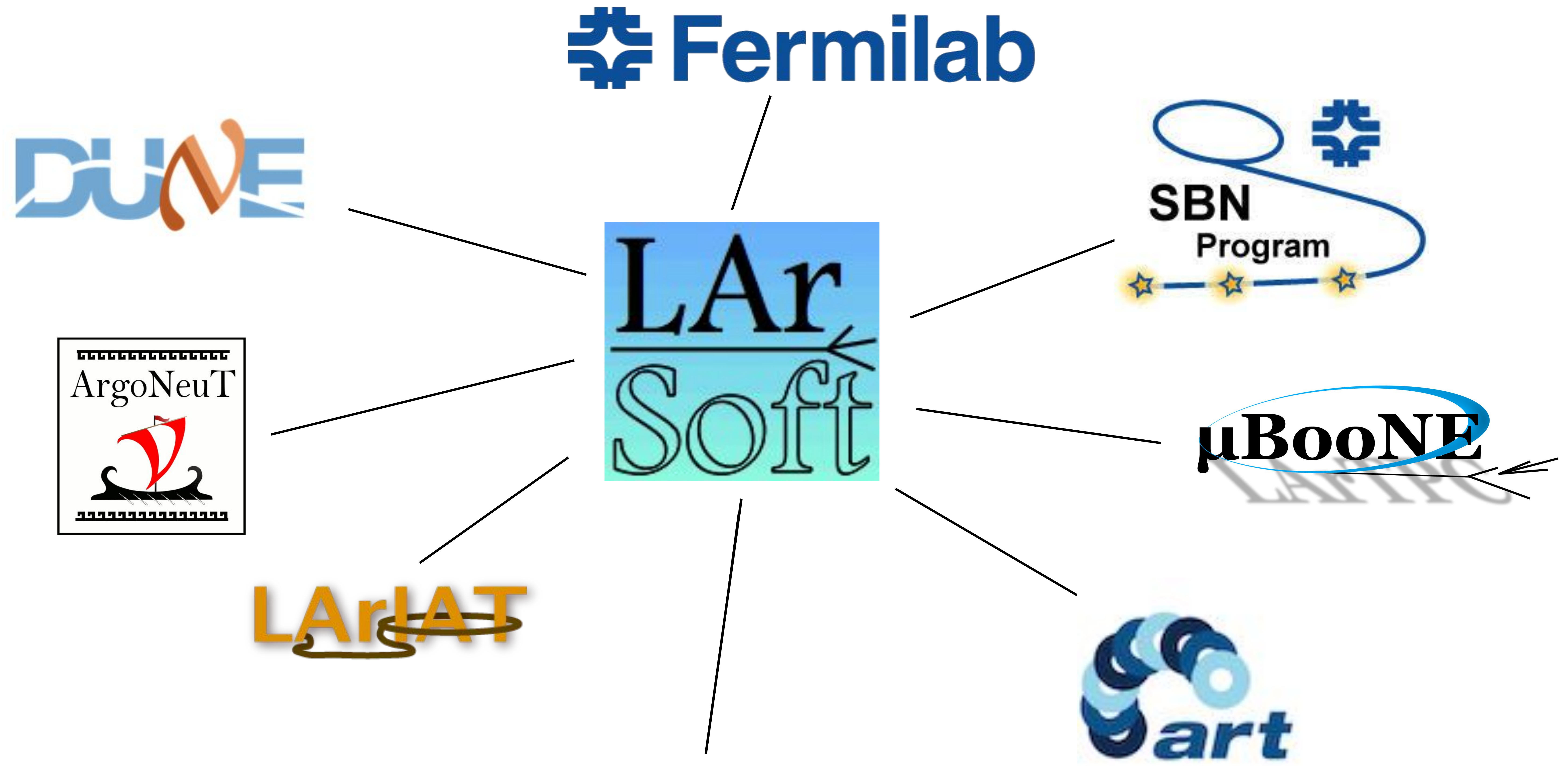
- Use common data structures and interfaces
- Express detector-specific differences via configuration
- Write algorithms that work for any / many LArTPCs

As a result, dramatically reduce the cost of developing this software for experiments that use LArTPC technology

The LArSoft Collaboration



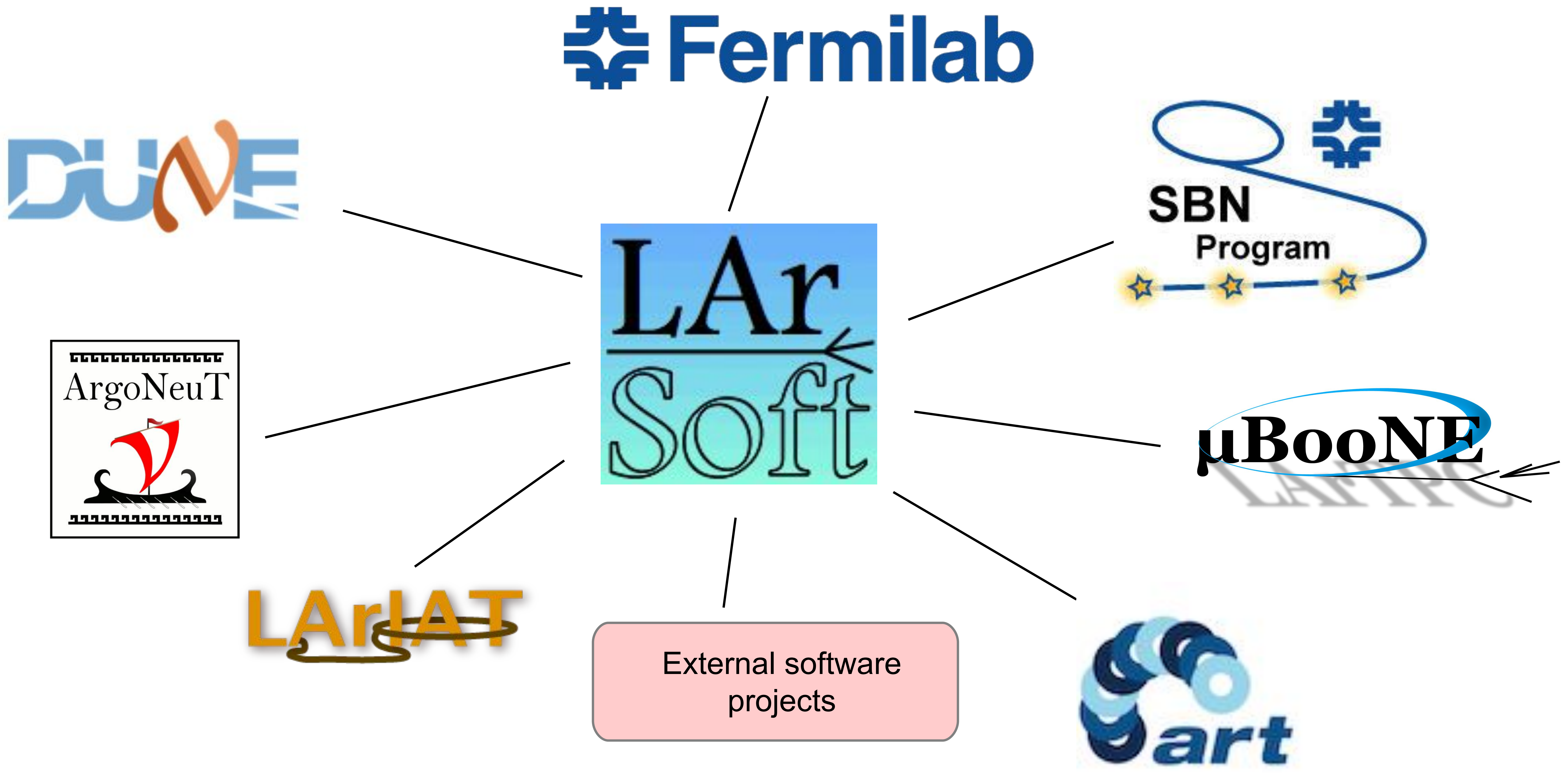
The LArSoft Collaboration



Sim: GENIE, Geant4, … Reco: Pandora, WireCell, … DL: TensorFlow, PyTorch, …

The LArSoft Collaboration

LArSoft is currently used in production by all these experiments



LArSoft design principles and coding practices

The basic philosophies and rules that underlie code sharing in core LArSoft code

1. Detector interoperability
2. Separation of framework and algorithm code
3. Use of standardized algorithm interfaces
4. Modularity
5. Design / write testable units of code
6. Document code in the source
7. Write code that is thread safe
8. Continuous integration

LArSoft design principles and coding practices

The basic philosophies and rules that underlie code sharing in core LArSoft code

1. Detector interoperability
2. Separation of framework and algorithm
3. Use of standardized algorithm interfaces
4. Modularity
5. Design / write testable units of code
6. Document code in the source
7. Write code that is thread safe
8. Continuous integration

- Critically important
- Allows use of LArSoft algorithm code outside of art, such as:
  - Lightweight analysis frameworks
    - Gallery, LArLite, ...
  - Specialized development / debugging environments
- Allows a future migration to another production framework, should that be needed

LArSoft design principles and coding practices

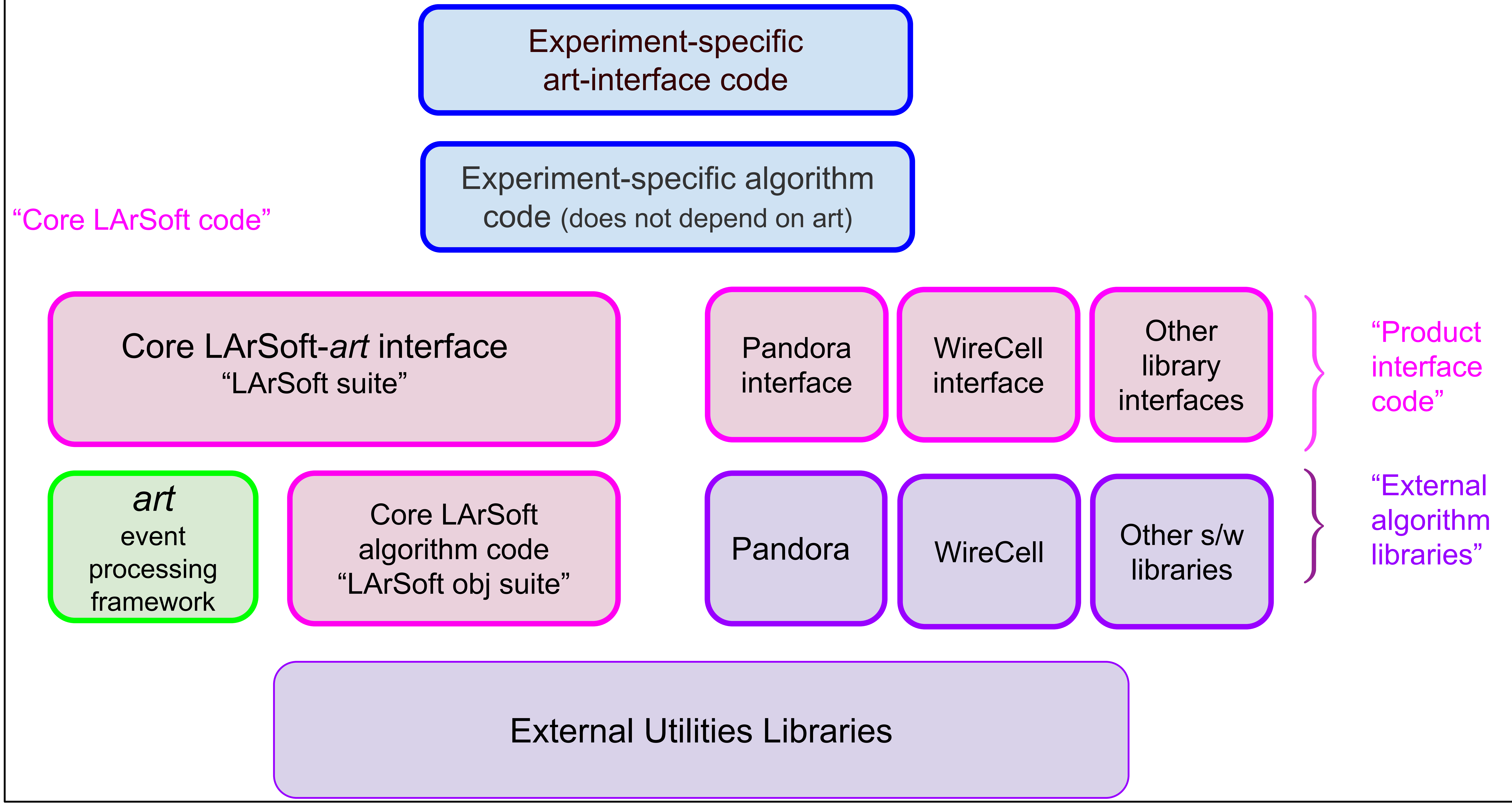
The basic philosophies and rules that underlie code sharing in core LArSoft code

1. Detector interoperability
2. Separation of framework and algorithm code
3. Use of standardized algorithm interfaces
4. Modularity
5. Design / write testable units of c
6. Document code in the source
7. Write code that is thread safe
8. Continuous integration

Provides a means to hide detector-specific details behind common interfaces

Also allows layering of algorithms to build sophistication

Structural components of LArSoft

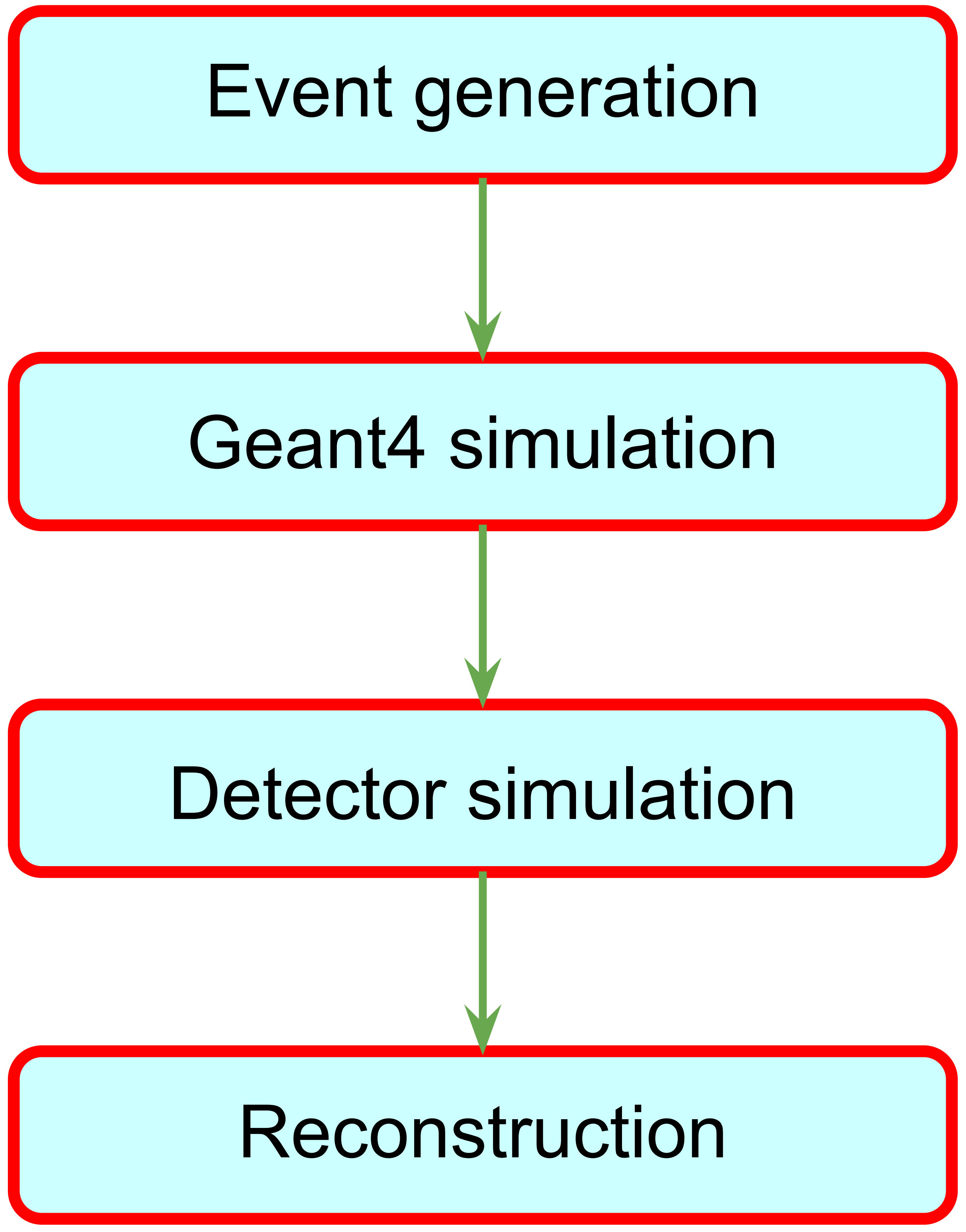


Experiment-specific art-interface code
Experiment-specific algorithm code (does not depend on art)
"Core LArSoft code"
Core LArSoft-*art* interface "LArSoft suite"
Pandora interface
WireCell interface
Other library interfaces
"Product interface code"
art event processing framework
Core LArSoft algorithm code "LArSoft obj suite"
Pandora
WireCell
Other s/w libraries
"External algorithm libraries"
External Utilities Libraries

What does LArSoft do? And what is in it?

Provides tools to carry out simulation, reconstruction and analysis of LArTPC data. (Note, analysis uses the output of any of the steps in the workflow, but a discussion of analysis is beyond the scope of this material.)

- Consider for instance, **an event generation, detector simulation, reconstruction workflow**



A general generation – simulation – reconstruction workflow

General generation-simulation-reconstruction workflow

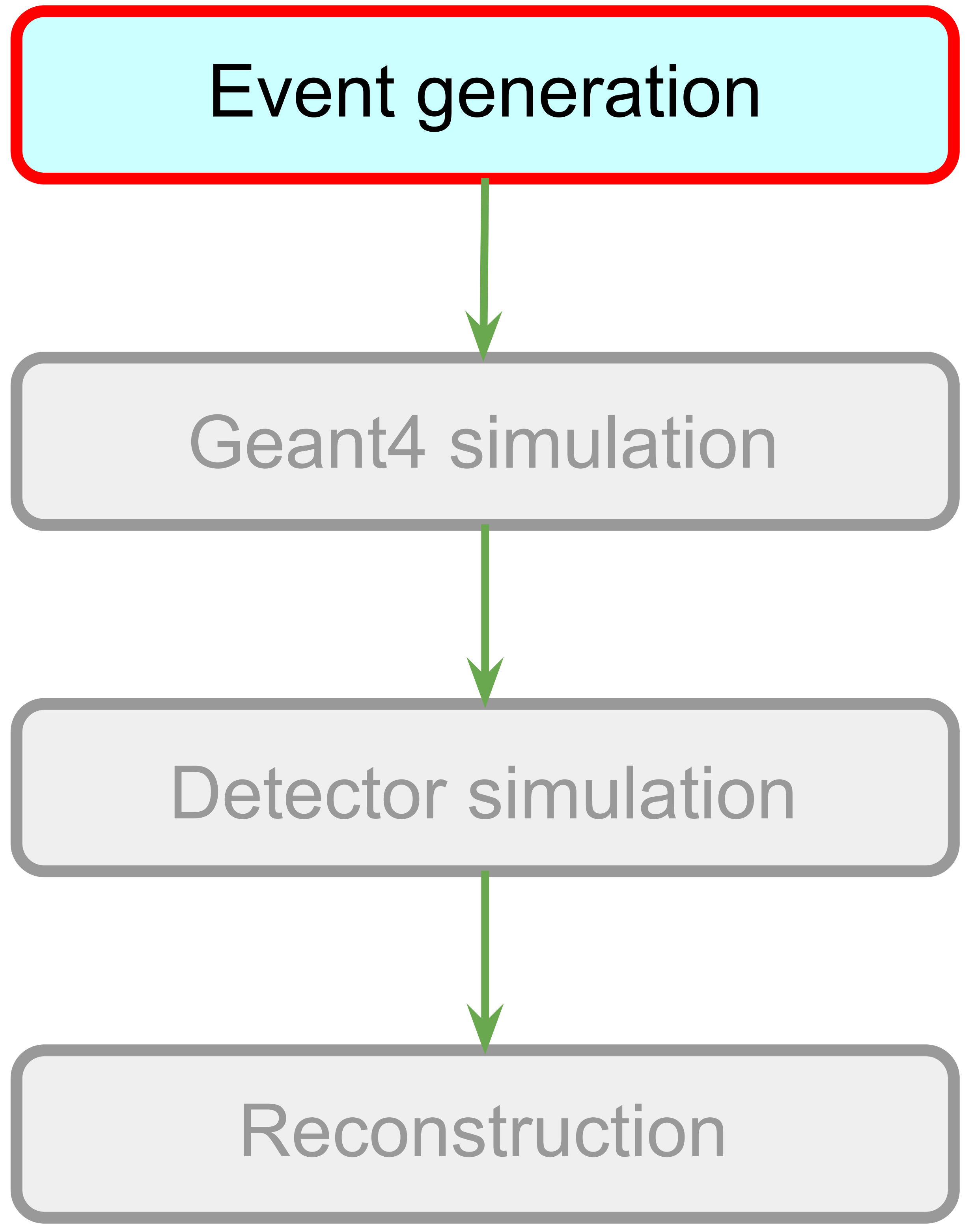
Event generators

- Genie: GENIEGen module
  - Interfaces to Genie neutrino event generator
  - larsim/larsim/EventGenerator/GENIE/
  - See genie.fcl in that directory
  - More documentation on the NuTools wiki page,
  - https://cdcvs.fnal.gov/redmine/projects/nutools/wiki

- Single particles: SingleGen module
  - larsim/larsim/EventGenerator

- Cosmic ray generators: CORSIKA, CRY
  - larsim/larsim/EventGenerator

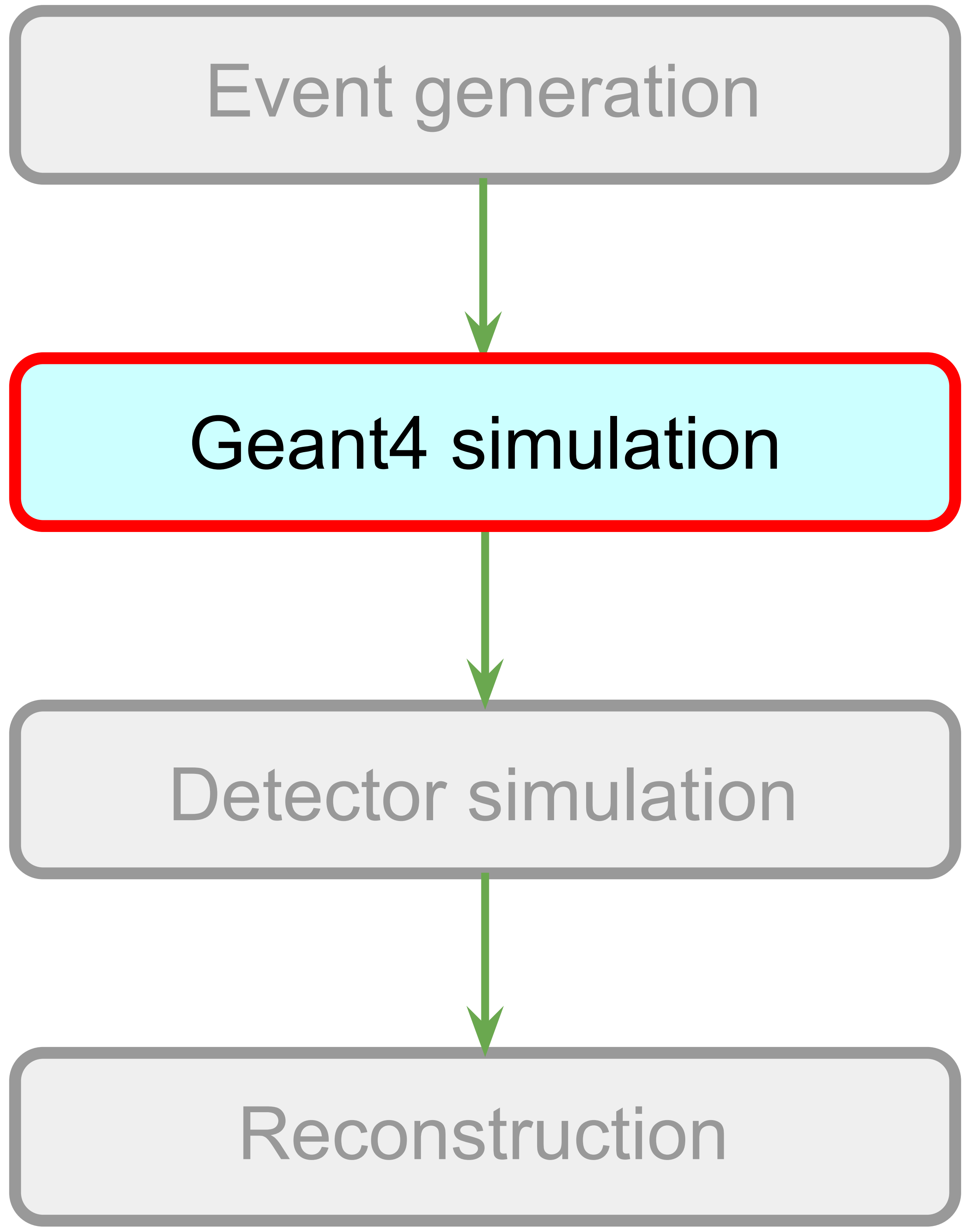
Others available via indirect common data exchange format, e.g., NuWro



General generation-simulation-reconstruction workflow

Geant4 detector simulation

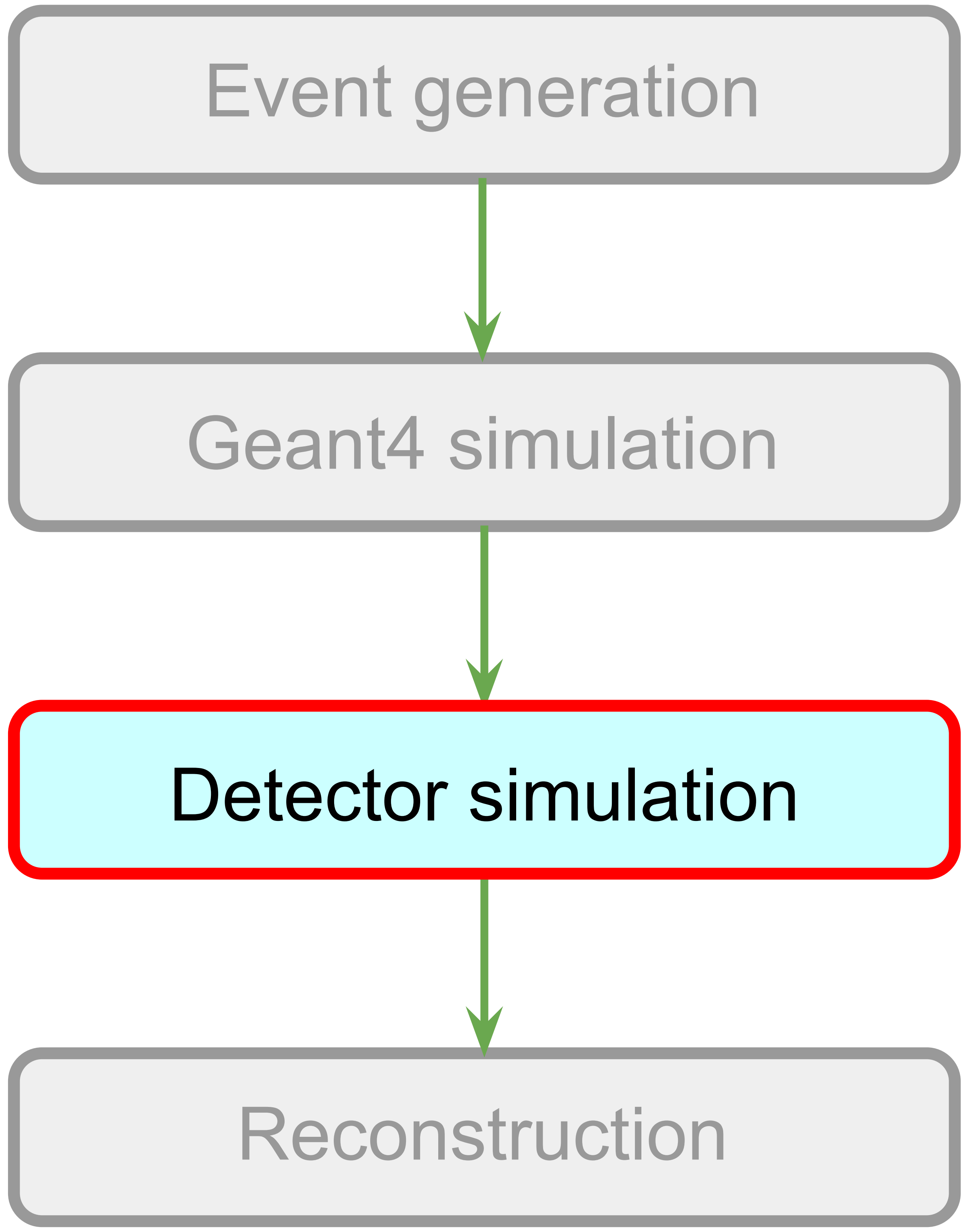
- Particle propagation simulation
- Models energy depositions in the detector
  - Rich, configurable models of particle interactions, optical properties (including detailed index of refraction, reflectivity, etc.)
  - Can perform optical simulation at single photon level
- The only simulation currently integrated with LArSoft



General generation-simulation-reconstruction workflow

A separate workflow in itself

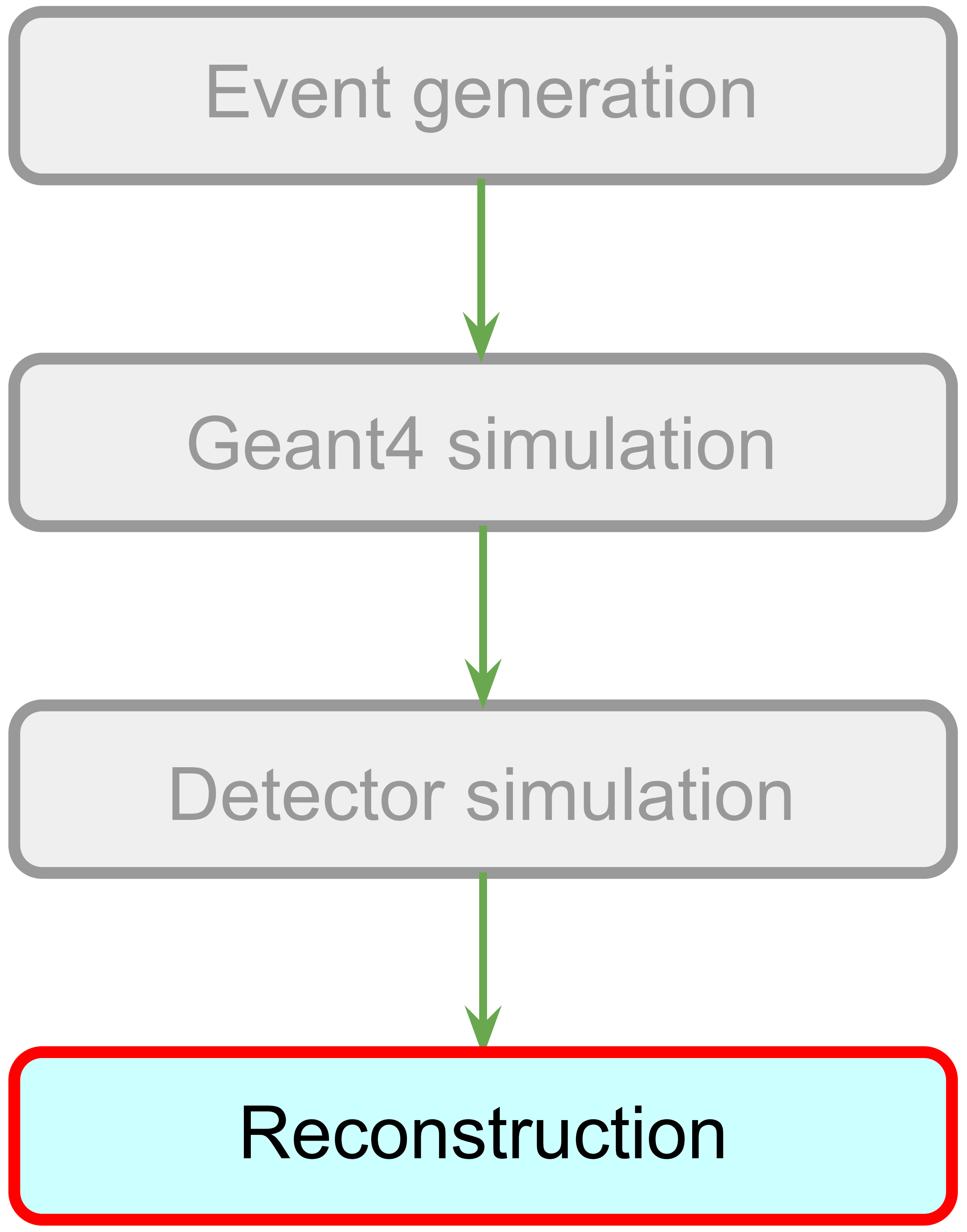
- Factorized into the following steps (implemented as separate modules / partly combined in WireCell)
  - Ionization and scintillation light modeling from energy depositions
  - Drift electron simulation
  - Anode region simulation, signal induction and noise modeling, digitization
  - Photon transport and detection model, including “S2 light” simulation for dual-phase detectors
  - Optical signal induction, noise modeling and digitization



General generation-simulation-reconstruction workflow

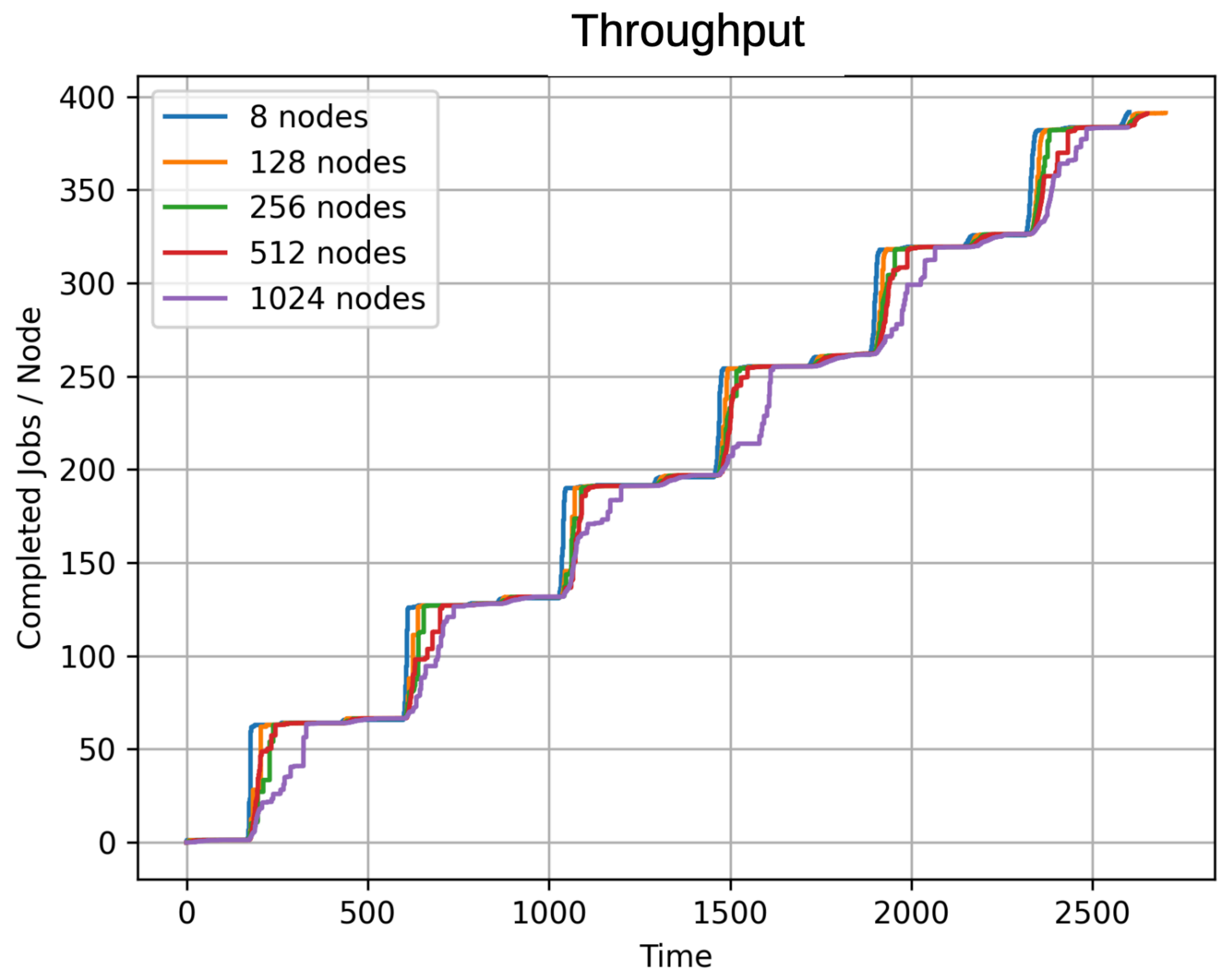
Three major paradigms, each with its own variants, modules, workflows

- 2D clustering and view matching
  - Pandora multi-algorithm approach
  - TrajCluster 2D
- Image processing / deep learning techniques
  - Pixel-level track/shower tagging from 2D images (code not yet fully available)
  - Hit-based track/shower discrimination
- 3D imaging
  - Wire-cell: tomographic charge matching across wire planes in time slices
  - TrajCluster3D / Cluster3D: time / charge matching across wire planes using hits.



LArSoft jobs at HPC

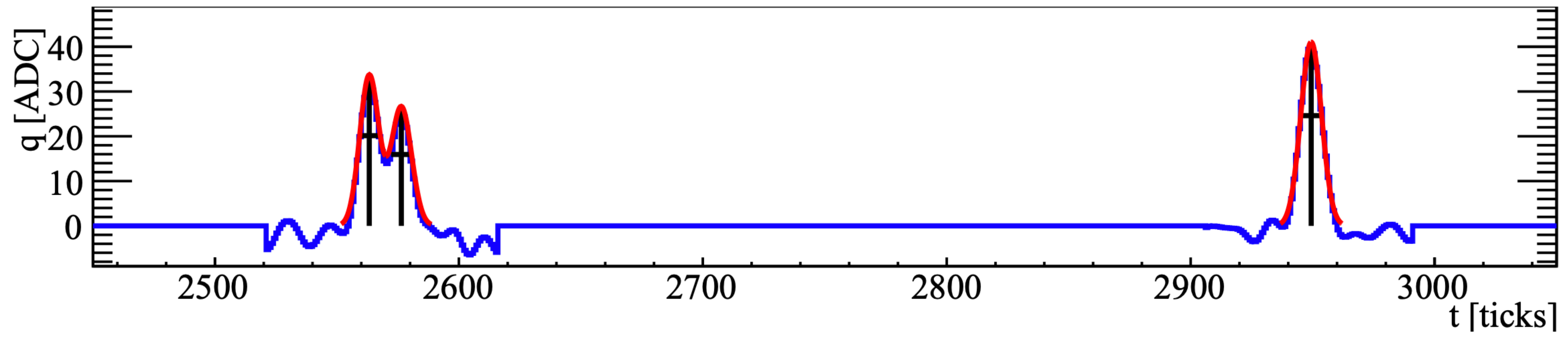
- Running LArSoft at large scale on Theta (~65,000 cores)
  - SBND and MicroBooNE
- Standard production chain
  - with only minor modifications to get running
  - but significant work to achieve scaling at node level
  - LArSoft run using standard releases in SL7 environment using Singularity containers
  - one job per core, no AVX-512



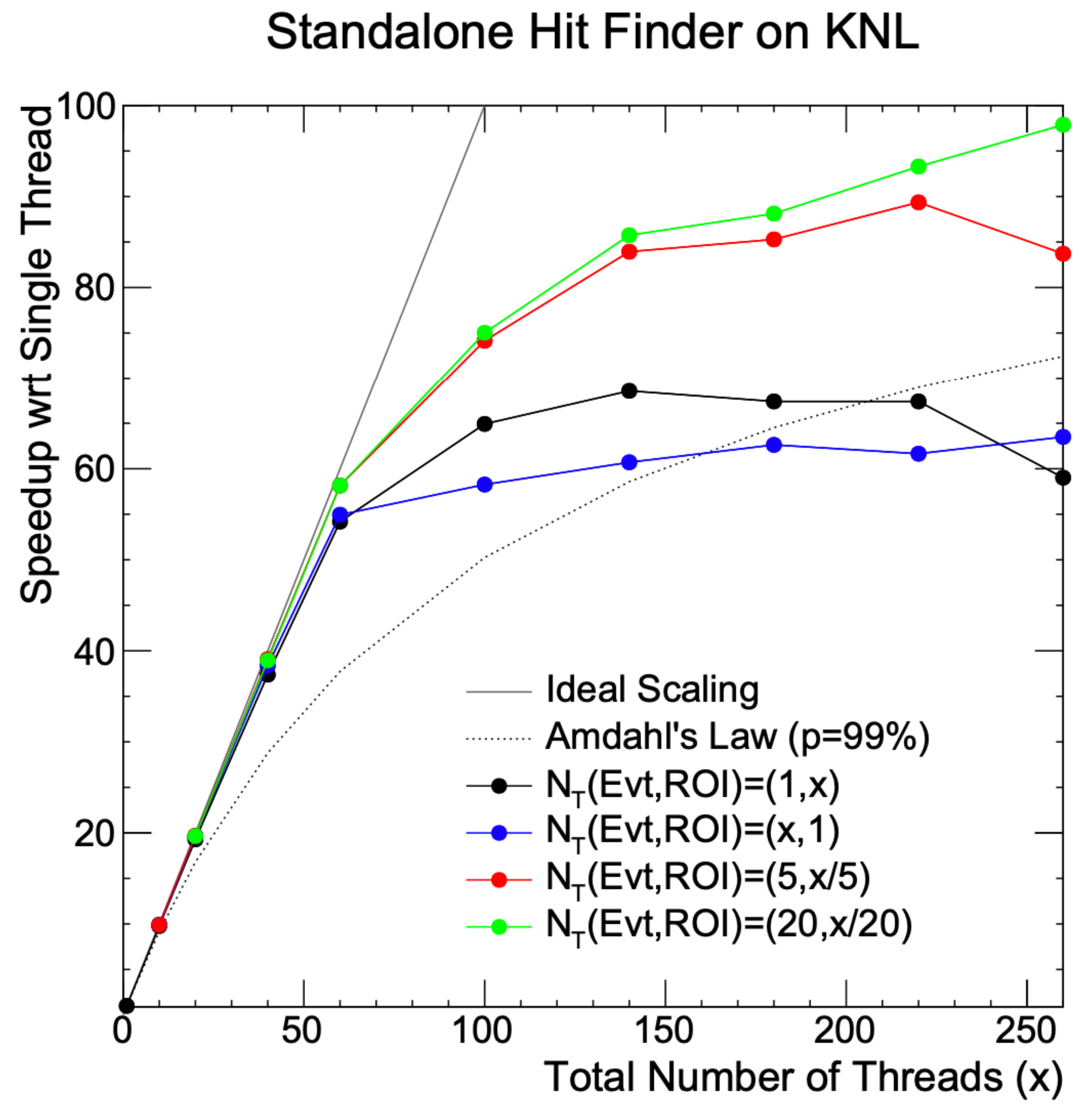
P. Green, Manchester

Parallelization and code optimization for HPC

- art framework provides multi-threading parallelization at event level with TBB
- Modules can implement parallelization at sub-event level
- Example: hit finder algorithm
  - vectorization over data points within minimization function: 2x speedup
  - multi-threading over wires
- Ongoing effort to natively compile LArSoft code on HPC systems using Spack
  - first successful tests on Theta, work ongoing to deploy dedicated workflow

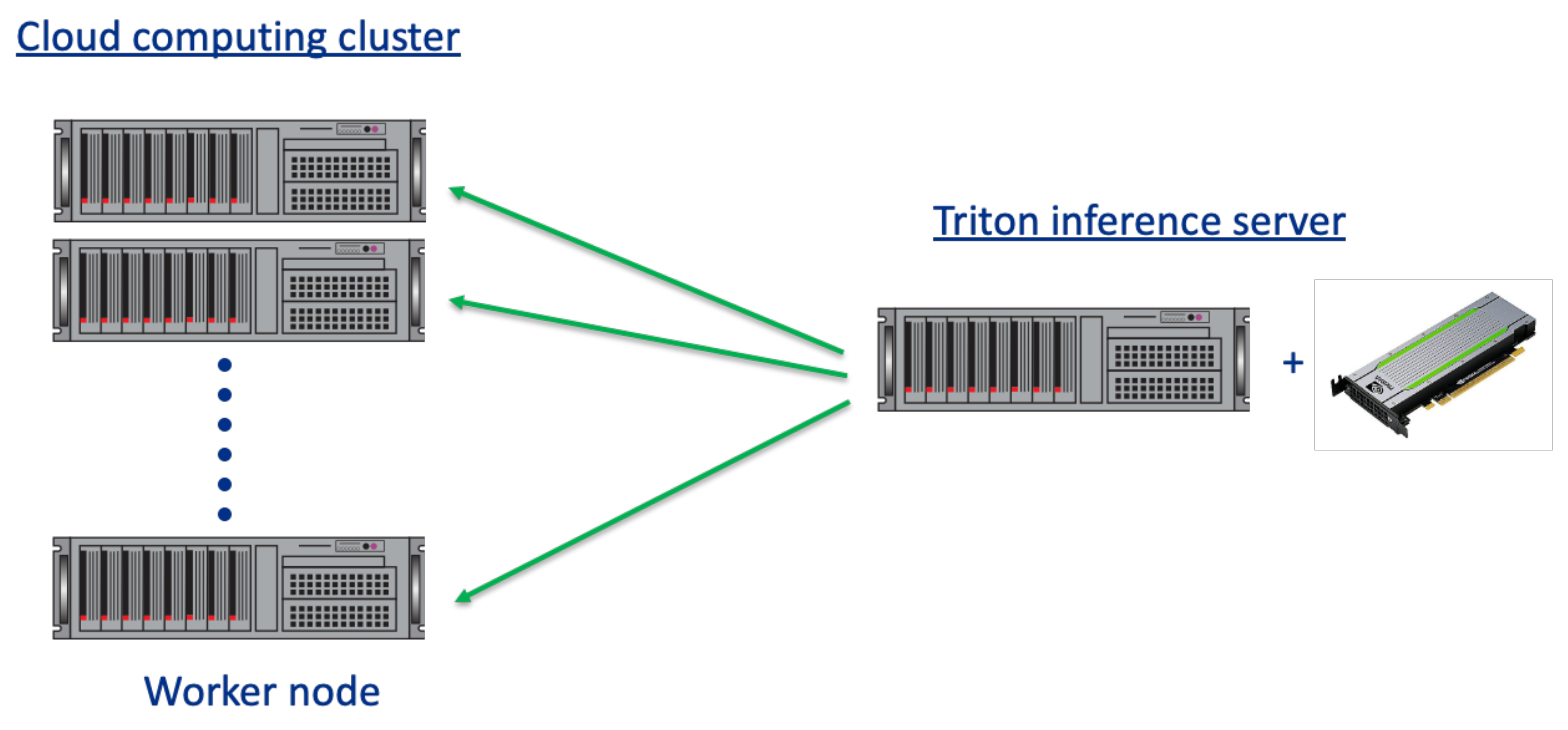


[S. Berkman et al, arXiv:2107.00812]()

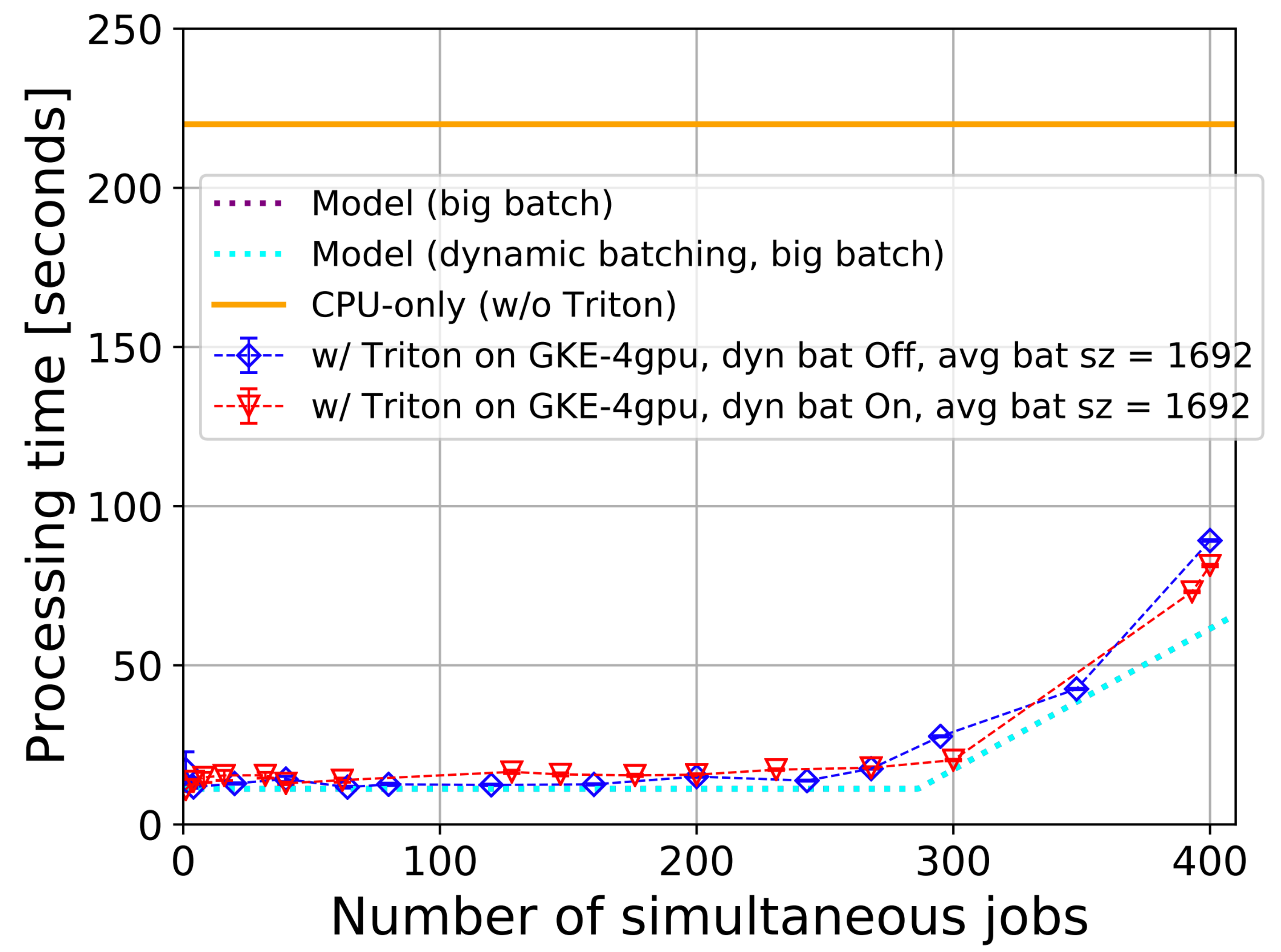


GPU as a Service

- LArTPC data is image-like: increasing usage of DL. Inference time on Grid/CPU is long, need to use GPU for quick processing
- Instead of buying GPU for each worker node, one can use "GPU as a Service" on a remote server
- Proof-of-principle: apply GPUaaS to the EMTrackMichelId module in the ProtoDUNE workflow.
  - This module uses a CNN to classify hits as being shower-like, tracklike, or Michel electron like.
  - NuSONIC inference client library deployed in LArSoft
  - see https://indico.fnal.gov/event/50528/



M. Wang et al., Front. Big Data 3 (2021) 48



ACTS

ACTS

- Website: acts.readthedocs.io

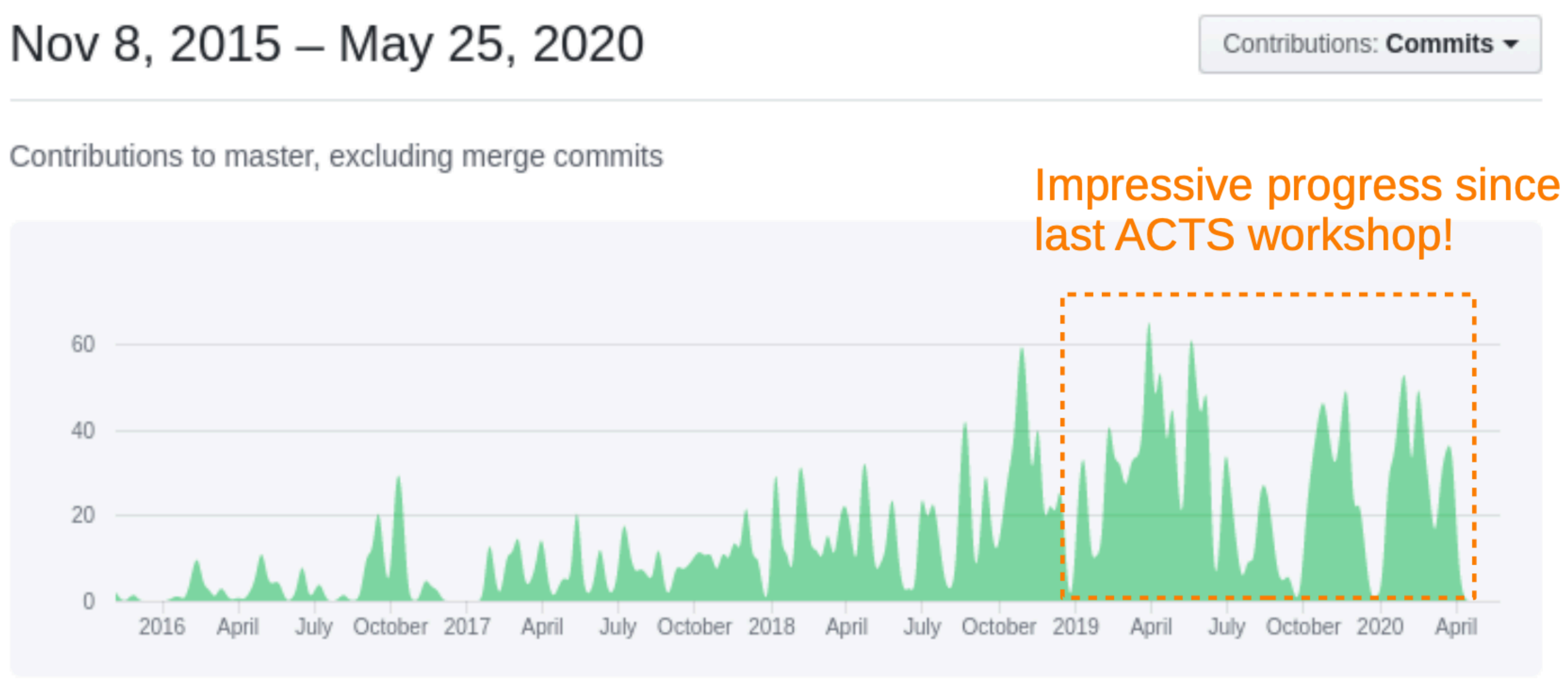
- Most of the content stolen from Xiaocong’s presentations in 2020
  - ACTS workshop: https://indico.cern.ch/event/917970/
    - https://indico.cern.ch/event/917970/contributions/3861752/attachments/2043758/3424088/ACTS_workshop_Xiaocong.pdf
  - CTD conference: https://indico.cern.ch/event/831165/
    - https://indico.cern.ch/event/831165/contributions/3717113/attachments/2024363/3396410/ACTS_CTD_Xiaocong.pdf

ACTS introduction

ACTS goals

- Prepare an experiment-independent tracking toolkit for future detectors, e.g. ATLAS at HL-LHC, based on ATLAS tracking experience
  - ATLAS tracking software is well tested but thread-unsafe and difficult to maintenable
- Provide an open-source R&D platform:
  - New tracking algorithms (see M. Kiehn's talk)
  - Hardware architectures (see G. Mania's talk)

Migration to github last month allowing contribution from people without CERN account



Nov 8, 2015 – May 25, 2020

Contributions: **Commits**

Contributions to master, excluding merge commits

Impressive progress since last ACTS workshop!

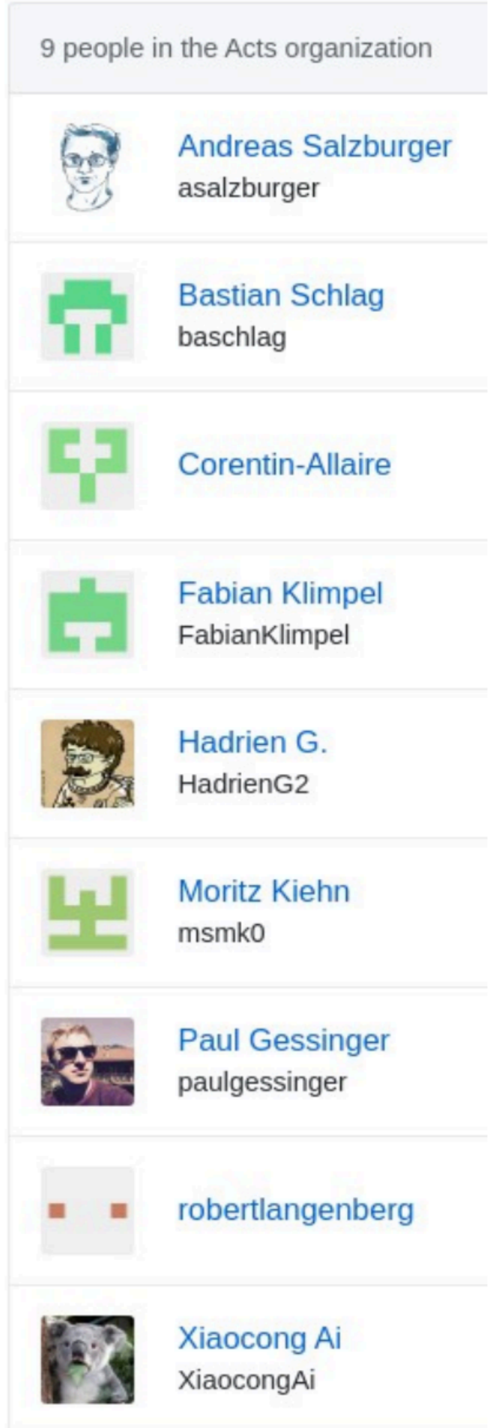
Jun 2016 - **v0.01.00**

Jan 2019 - **v0.08.00**

May 2020- **v0.25.00**

https://github.com/acts-project/acts

An incomplete list of ACTS developers



9 people in the Acts organization

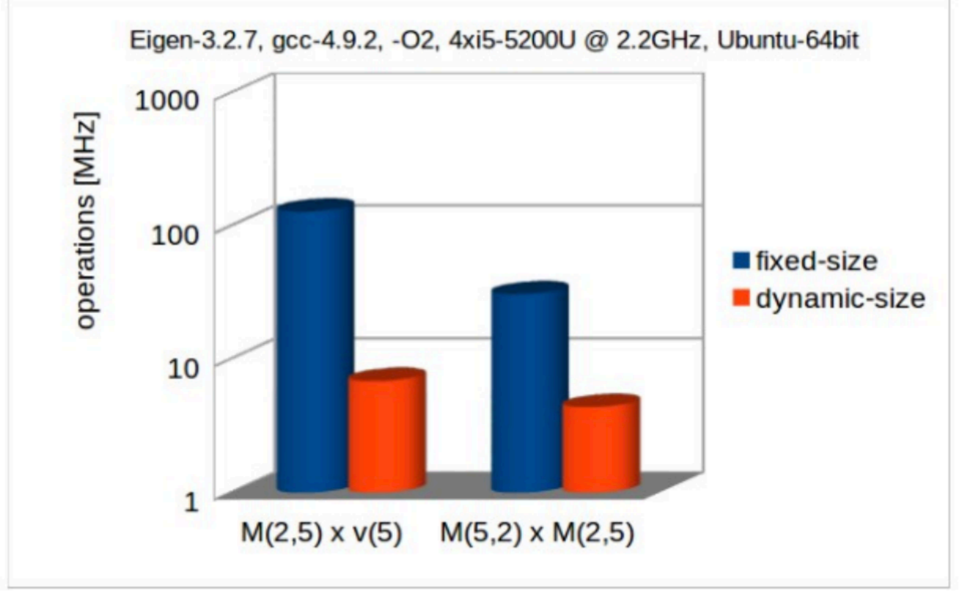
Andreas Salzburger	asalzburger
Bastian Schlag	baschlag
Corentin-Allaire	
Fabian Klimpel	FabianKlimpel
Hadrien G.	HadrienG2
Moritz Kiehn	msmk0
Paul Gessinger	paulgessinger
robertlangenberg	
Xiaocong Ai	XiaocongAi

3

ACTS Design and Code Structure

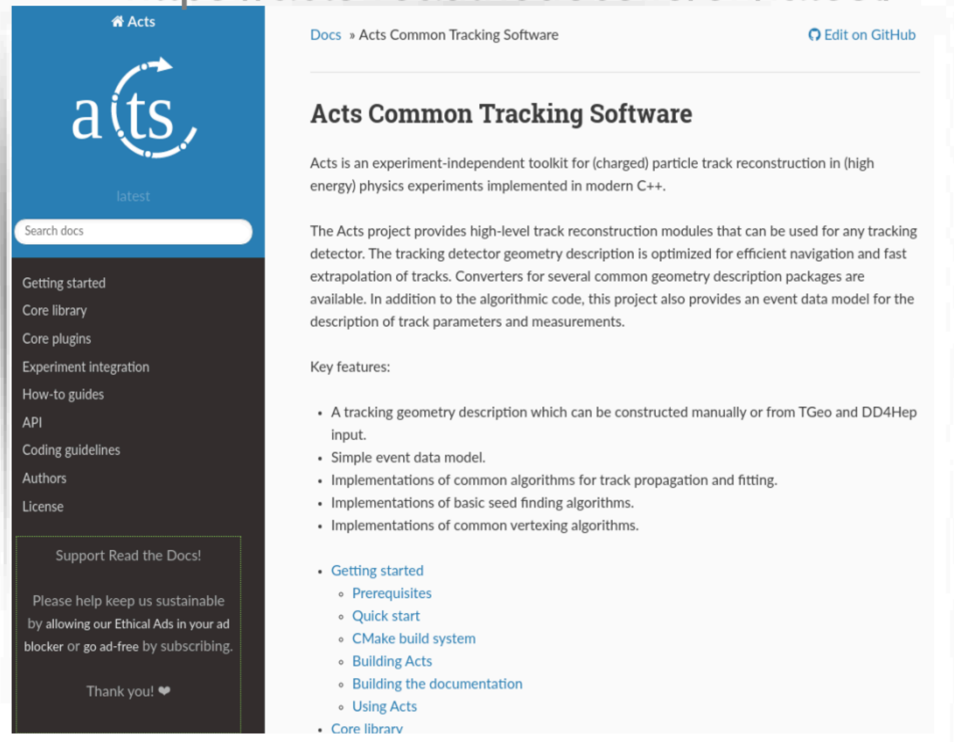
ACTS design

- Modern C++ 17 concepts
- Highly-templated design to avoid virtual lookup
  - Detector and magnetic field agnostic
- Strict thread-safety to facilitate concurrency
  - Const-correctness, stateless tools
- Efficient memory allocation and access
  - Eigen-based Event Data Model (EDM)
- Supports for contextual condition data
  - Geometry/Calibration/Magnetic field
- Rigorous unit tests
- Highly configurable for usability
- Minimal dependencies
- Well-documented



```
size_t algorithmNumber;              ///< Unique algorithm identifier
size_t eventNumber;                  ///< Unique event identifier
WhiteBoard& eventStore;              ///< Per-event data store
Acts::GeometryContext geoContext;    ///< Per-event geometry context
Acts::MagneticFieldContext
    magFieldContext;                      ///< Per-event magnetic Field contex
Acts::CalibrationContext calibContext;    ///< Per-event calibration context
```

https://acts.readthedocs.io/en/latest/



Acts

latest

Search docs

Getting started
Core library
Core plugins
Experiment integration
How-to guides
API
Coding guidelines
Authors
License

Support Read the Docs!

Please help keep us sustainable by allowing our Ethical Ads in your ad blocker or go ad-free by subscribing.

Thank you! ❤

Docs » Acts Common Tracking Software   Edit on GitHub

Acts Common Tracking Software

Acts is an experiment-independent toolkit for (charged) particle track reconstruction in (high energy) physics experiments implemented in modern C++.

The Acts project provides high-level track reconstruction modules that can be used for any tracking detector. The tracking detector geometry description is optimized for efficient navigation and fast extrapolation of tracks. Converters for several common geometry description packages are available. In addition to the algorithmic code, this project also provides an event data model for the description of track parameters and measurements.

Key features:

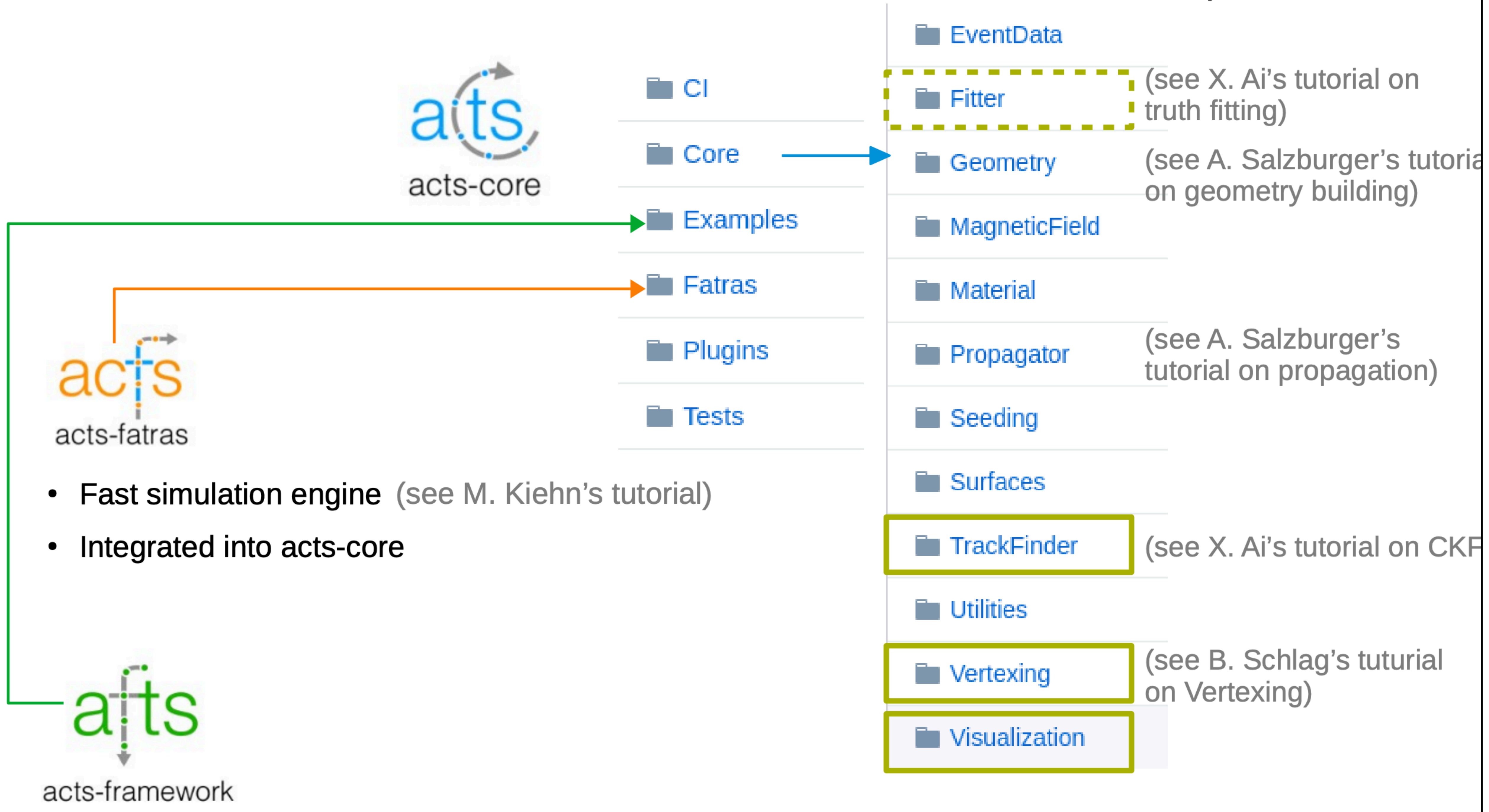
- A tracking geometry description which can be constructed manually or from TGeo and DD4Hep input.
- Simple event data model.
- Implementations of common algorithms for track propagation and fitting.
- Implementations of basic seed finding algorithms.
- Implementations of common vertexing algorithms.

- Getting started
  - Prerequisites
  - Quick start
  - CMake build system
  - Building Acts
  - Building the documentation
  - Using Acts
- Core library

4

ACTS components and functionalities

Infrastructure consolidation and new features since last ACTS workshop



acts-core
CI
Core
Examples
Fatras
Plugins
Tests
EventData
Fitter
(see X. Ai's tutorial on truth fitting)
Geometry
(see A. Salzburger's tutoria on geometry building)
MagneticField
Material
Propagator
(see A. Salzburger's tutorial on propagation)
Seeding
Surfaces
TrackFinder
(see X. Ai's tutorial on CKF
Utilities
Vertexing
(see B. Schlag's tuturial on Vertexing)
Visualization
acts-fatras
acts-framework

- Fast simulation engine (see M. Kiehn's tutorial)
- Integrated into acts-core

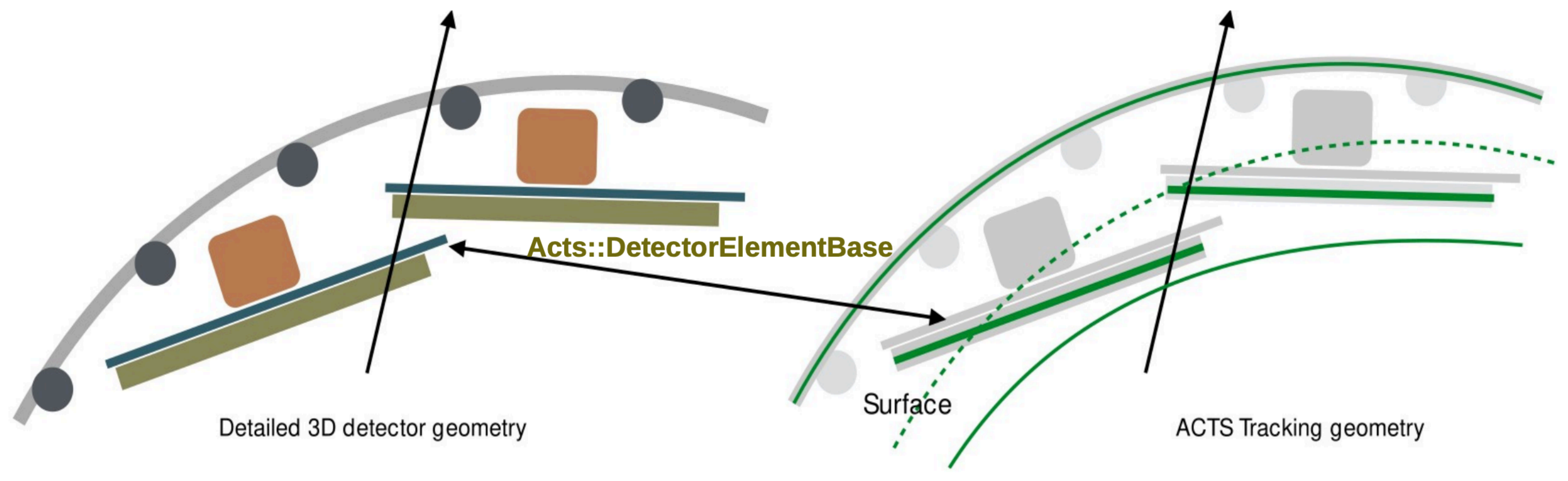
- A light-weight Gaudi style test framework for event processing, integration and concurrency test
- Integration into acts-core to ease maintenance and performance monitoring

5

Detector Geometry and Material

Geometry

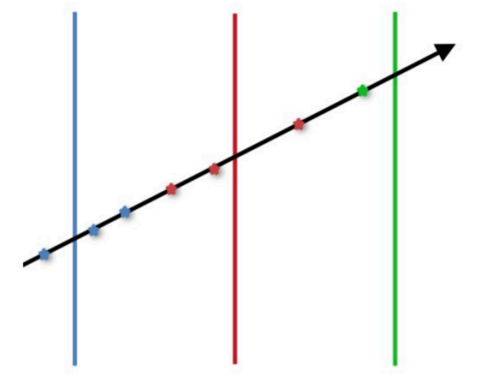
- To reduce CPU consumption and navigation speed-up, tracking geometry (i.e. geometry used for track reconstruction) is simplified from full simulation geometry
  - Binding via Acts::DetectorElementBase which can be converted from other detector element representation via geometry plugins:
    - DD4hep, TGeo, GeoModel Plugin
- Implemented HEP detector geometry
  - Silicon, Calorimeter, MuonSpectrometer



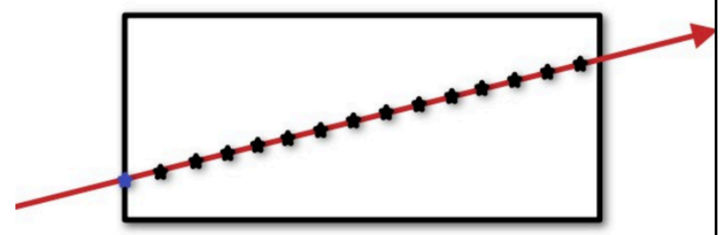
6

Material description

- Material effects need to be considered in tracking
- Material mapping tools (see C. Allaire's slides) are available to map Gean4-based full detector material (recorded using Geantino scan) onto surfaces or volumes:
  - Discrete binned surfaces (for e.g. Silicon detector)
  - 3D volume grid points (for e.g. Calorimeter)

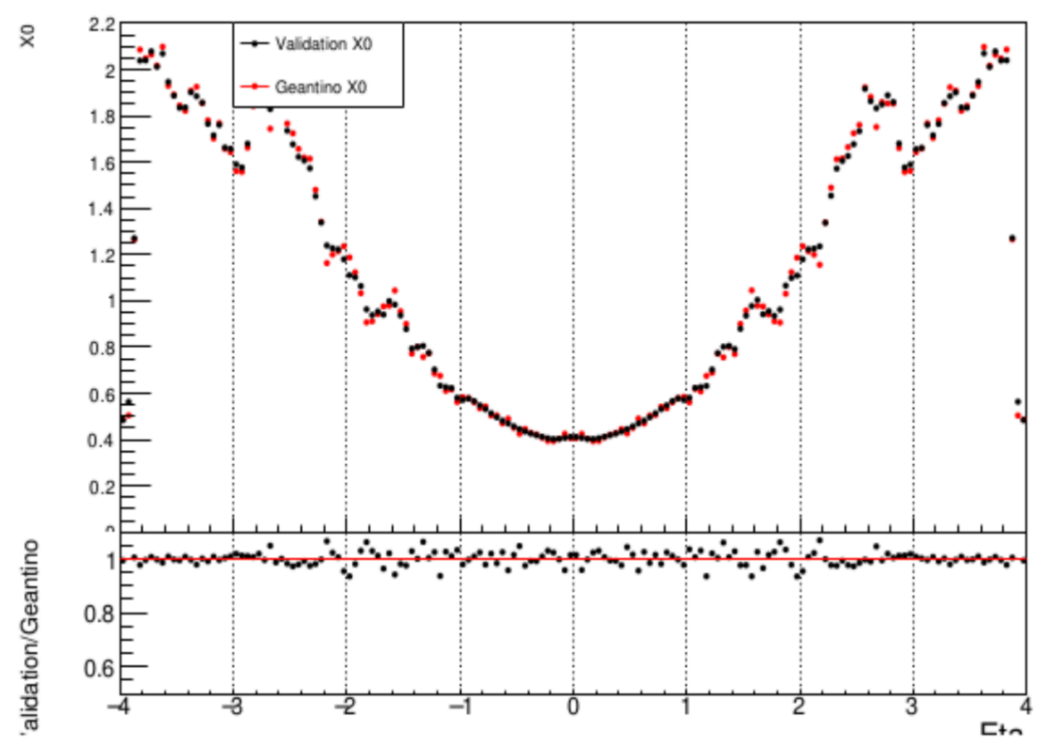


Surface Material Mapping

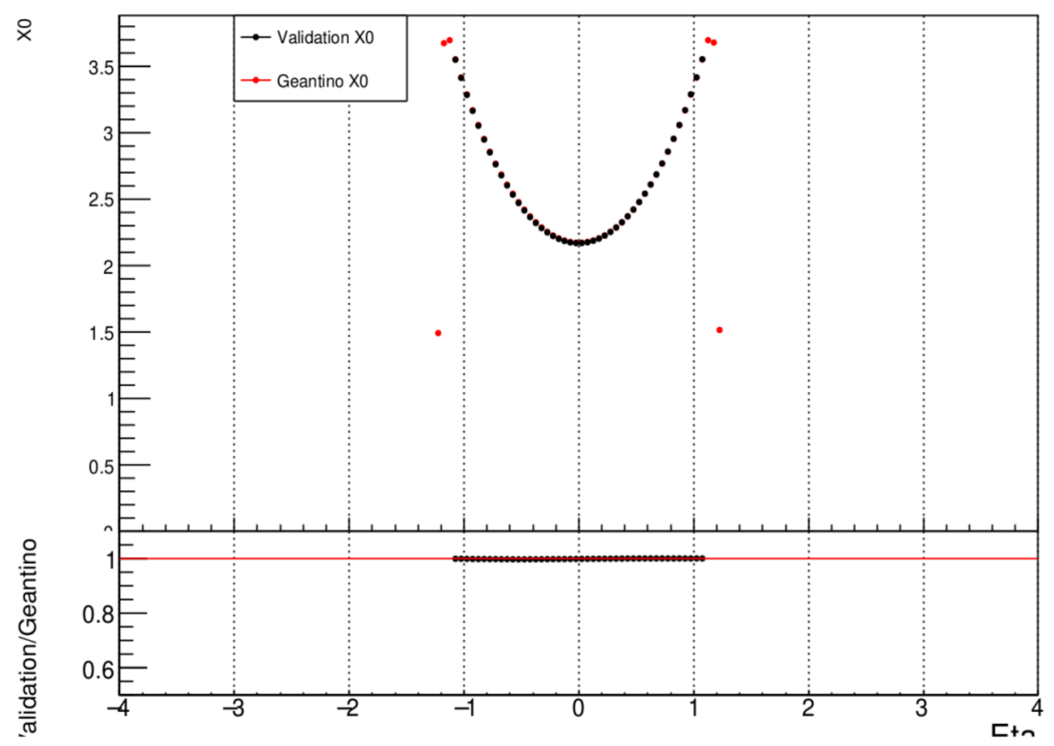


Volume Material Mapping

X0 ratio Validation/Geantino vs η for ITk



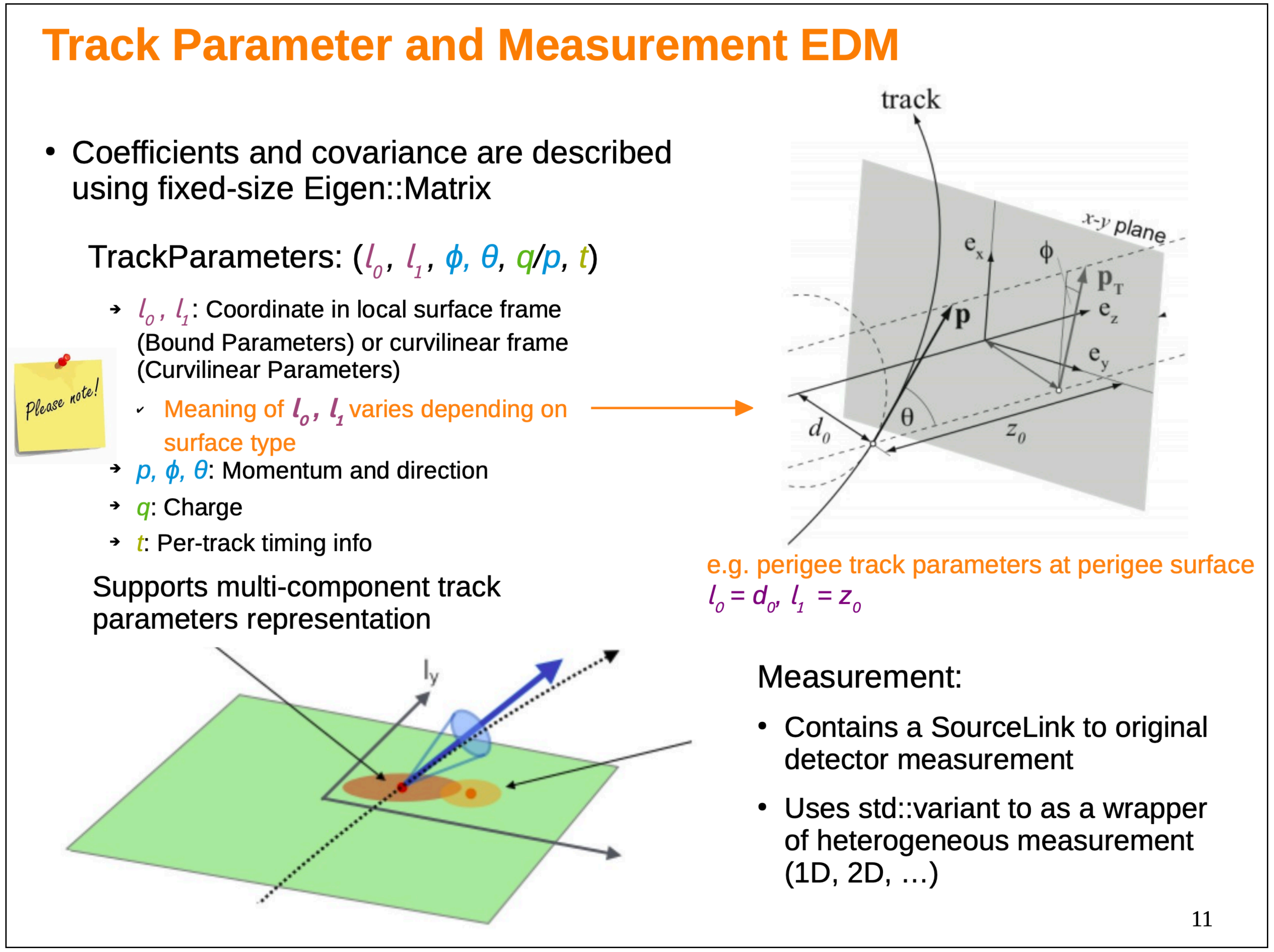
X0 ratio Validation/Geantino vs h for a dummy Calorimeter



8

Fermilab

Kalman Filter Track Fitting



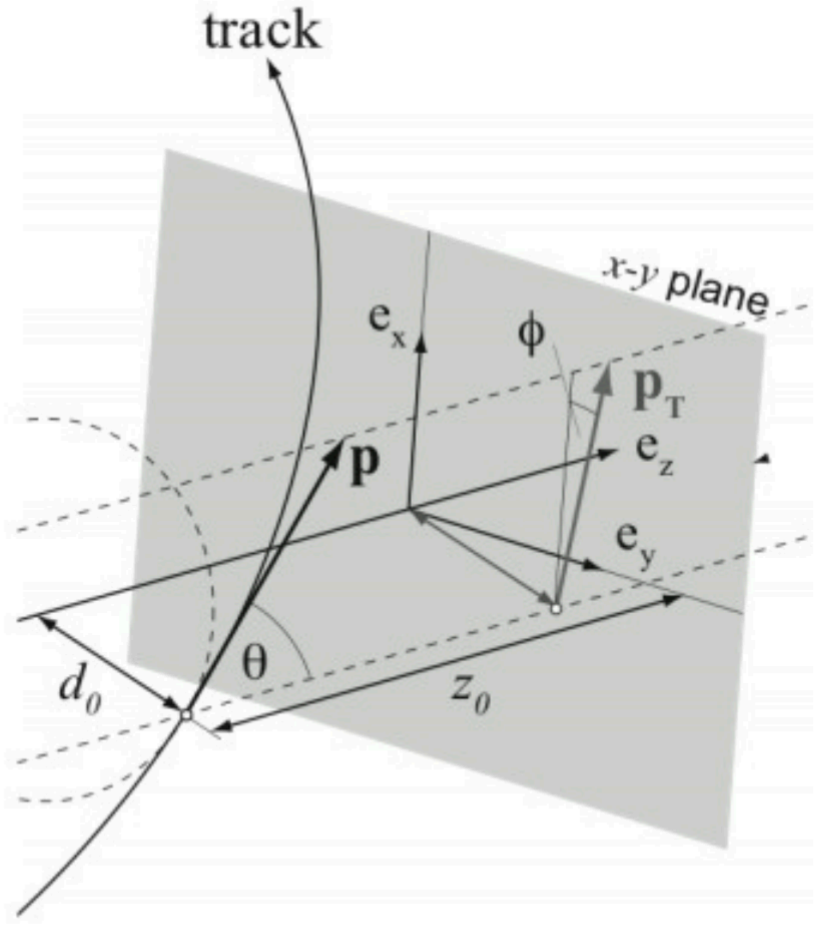
Track Parameter and Measurement EDM

- Coefficients and covariance are described using fixed-size Eigen::Matrix

TrackParameters: (l_0, l_1, ϕ, θ, q/p, t)

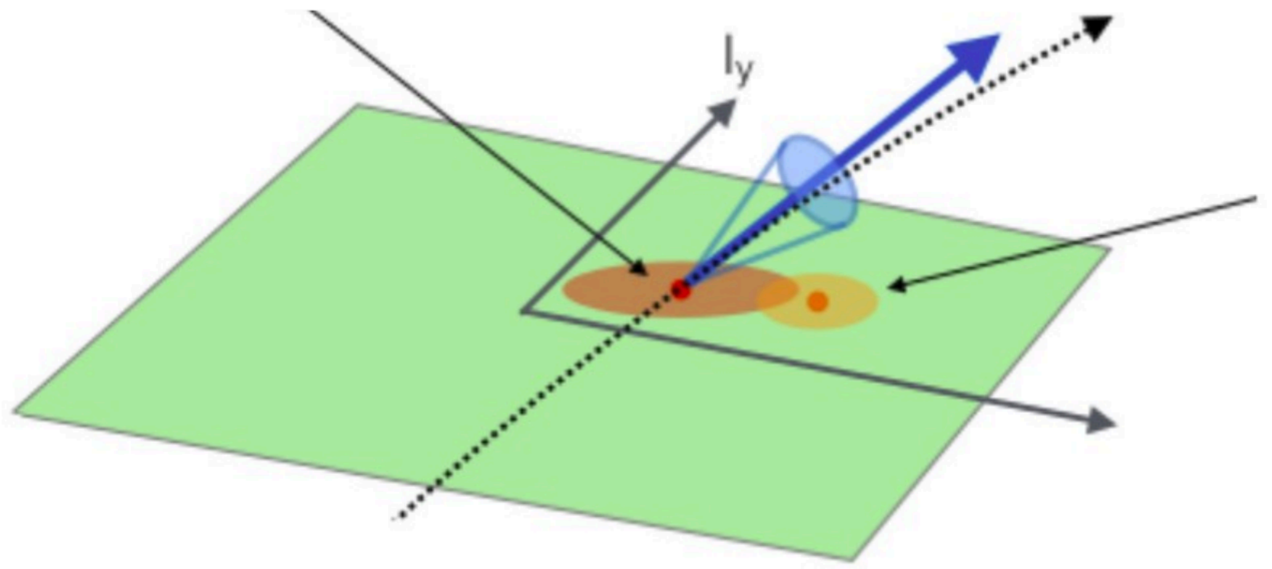
- l_0, l_1: Coordinate in local surface frame (Bound Parameters) or curvilinear frame (Curvilinear Parameters)
  - ✓ Meaning of l_0, l_1 varies depending on surface type
- p, ϕ, θ: Momentum and direction
- q: Charge
- t: Per-track timing info

Please note!



e.g. perigee track parameters at perigee surface
$l_0 = d_0$, $l_1 = z_0$

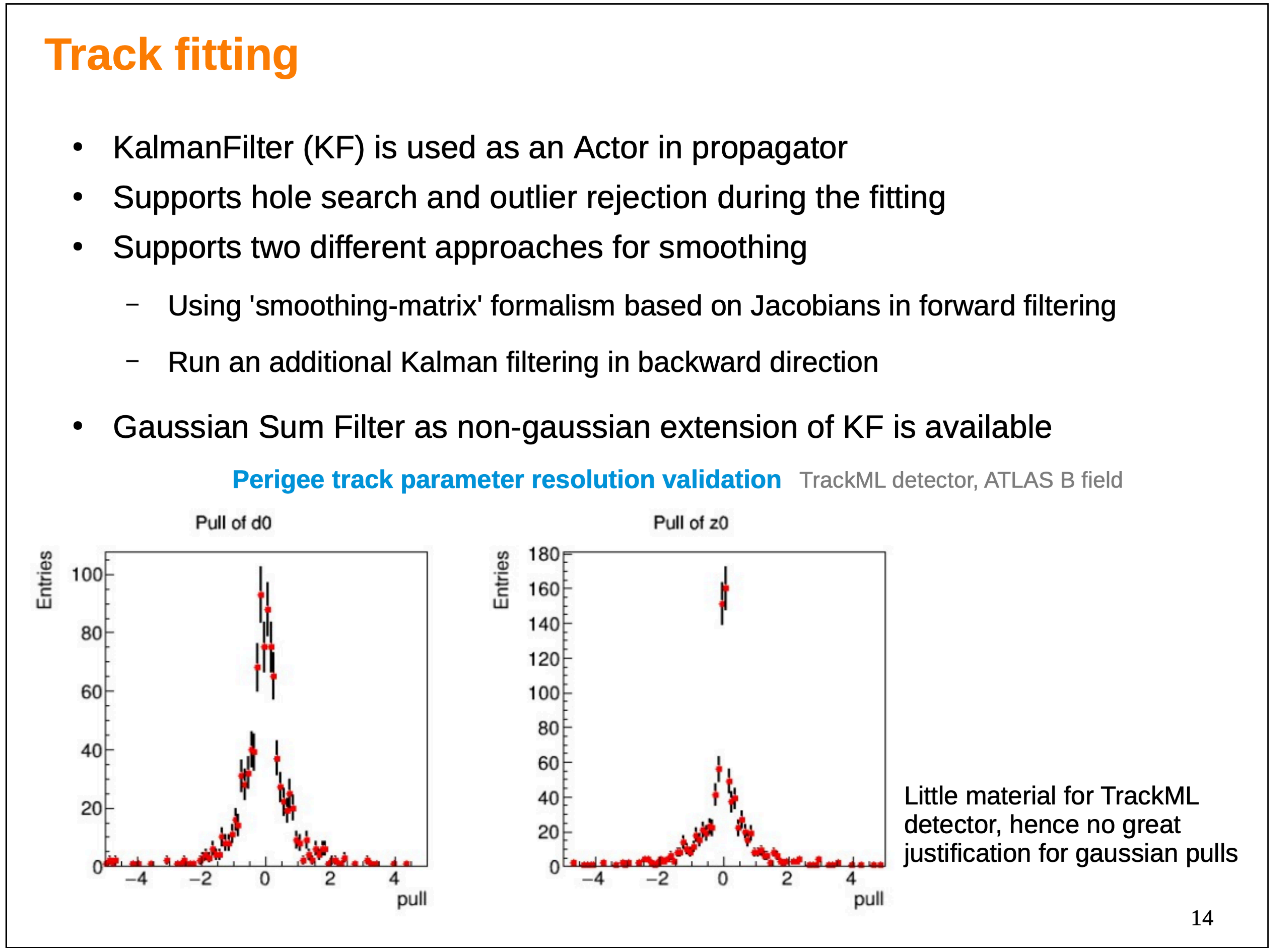
Supports multi-component track parameters representation



Measurement:

- Contains a SourceLink to original detector measurement
- Uses std::variant to as a wrapper of heterogeneous measurement (1D, 2D, …)

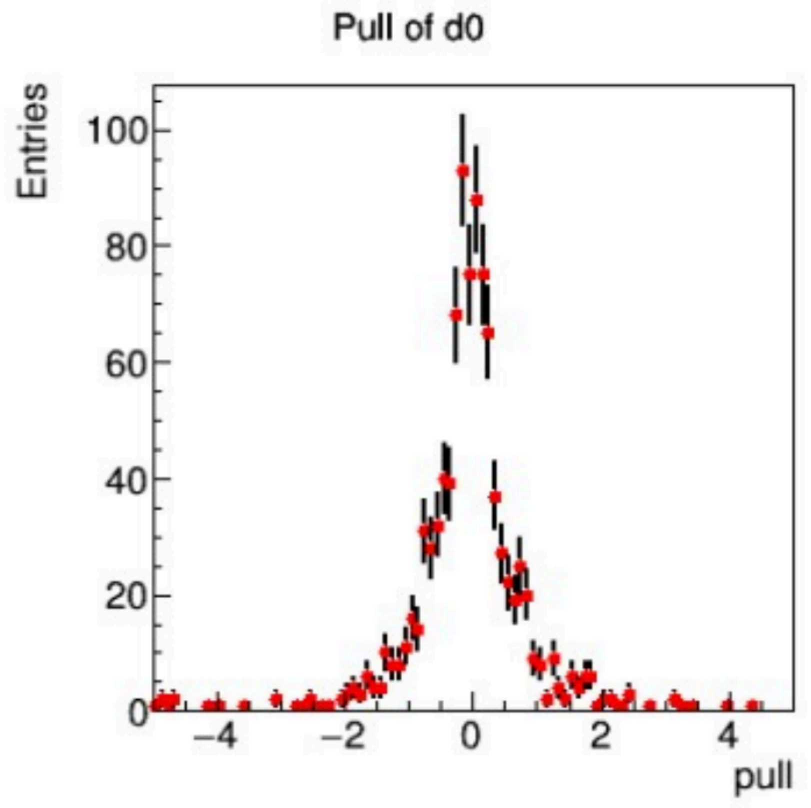
11

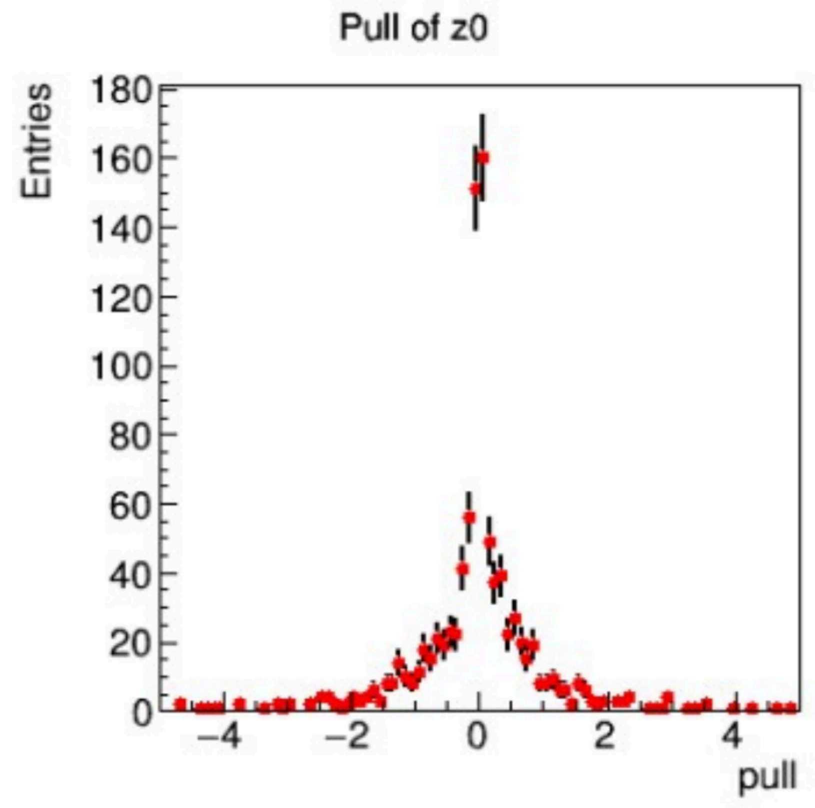


Track fitting

- KalmanFilter (KF) is used as an Actor in propagator
- Supports hole search and outlier rejection during the fitting
- Supports two different approaches for smoothing
  - Using 'smoothing-matrix' formalism based on Jacobians in forward filtering
  - Run an additional Kalman filtering in backward direction
- Gaussian Sum Filter as non-gaussian extension of KF is available

Perigee track parameter resolution validation TrackML detector, ATLAS B field





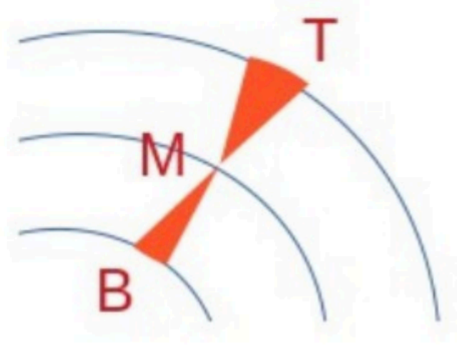
Little material for TrackML detector, hence no great justification for gaussian pulls

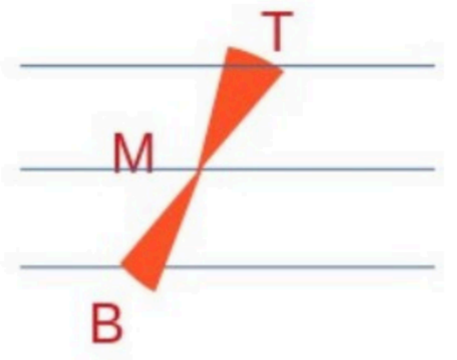
14

Kalman Filter Track Finding

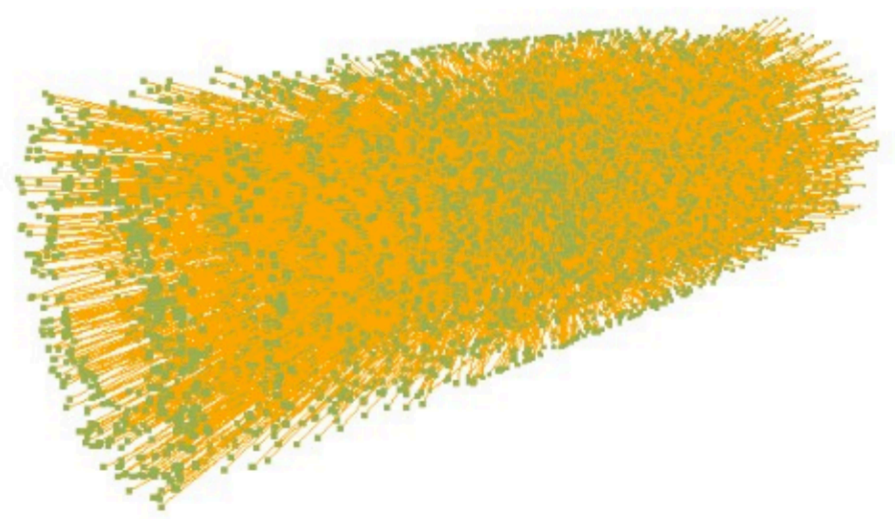
Track finding

- A combinatorial seed finder for track seeding
  - Fine-grained parallelism (independent search of Top and Bottom SpacePoint for Middle SP)





- The Combinatorial Kalman Filter (CKF) for track following
  - Simultaneous tracking fitting and finding (no refitting is needed)
  - Allows track branching if more than one compatible measurement found on a surface
    - Supports user-defined measurement search and branching strategy
      - Default selection criteria is based on Kalman filtering χ^2
  - Allows stopping of bad quality branch



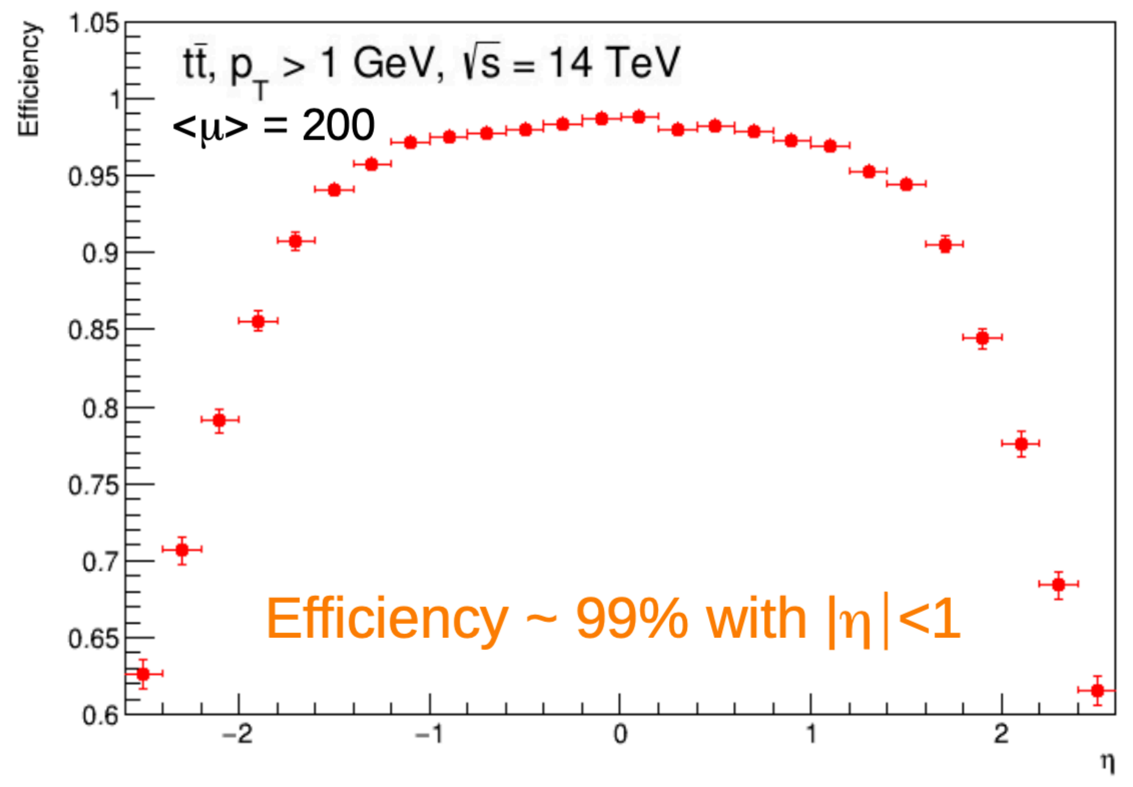
CKF results for ttbar events with μ = 200 (~7k particles, ~80k hits)

CKF performance

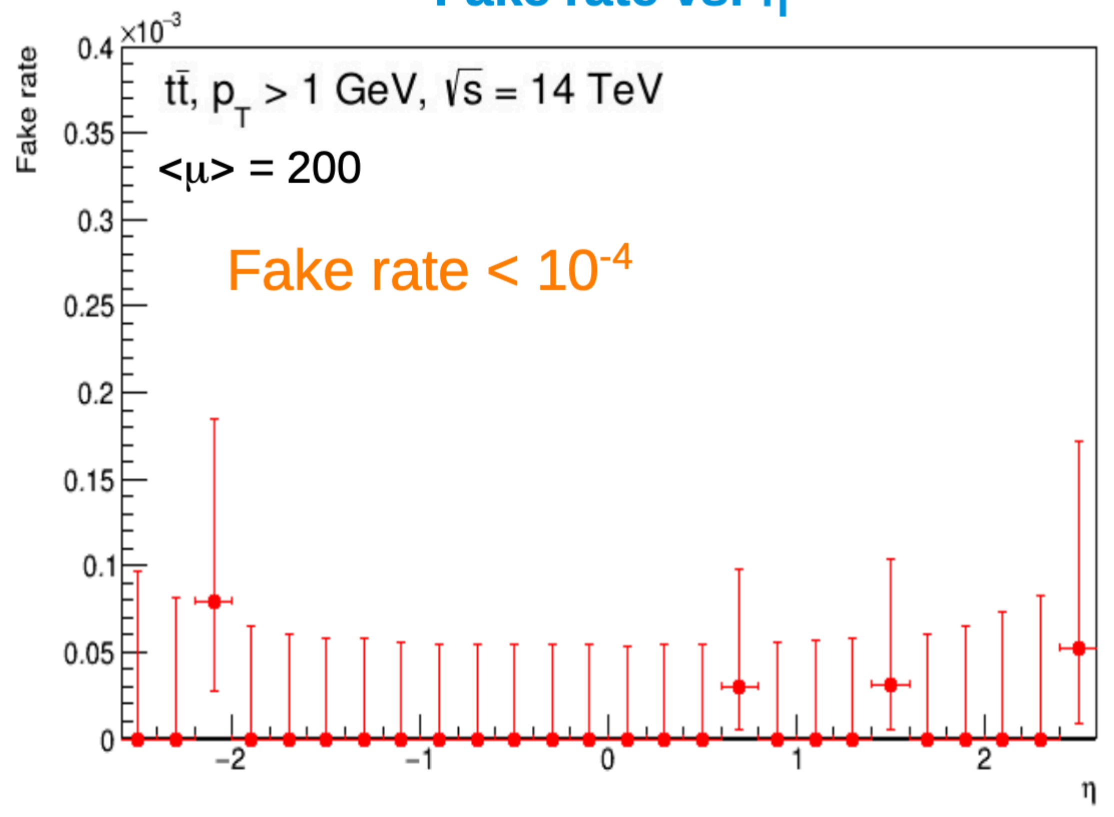
TrackML detector, ATLAS B field

- All hits from truth particles with p_T>100 MeV are considered
  - Track finding efficiency: $\frac{N_{reco}(selected, matched)}{N_{truth}(selected)}$
  - Fake rate: $\frac{N_{reco}(selected, unmatched)}{N_{reco}(selected)}$
  - Duplication rate: $\frac{N_{reco}(selected, matched, duplicated)}{N_{reco}(selected, matched)}$
    - ➔ Reco-truth matching: $\frac{N_{hits}(Majority)}{N_{hits}(Total)} > 0.5$
    - ➔ Simple track selection: n_{Hits}>=9

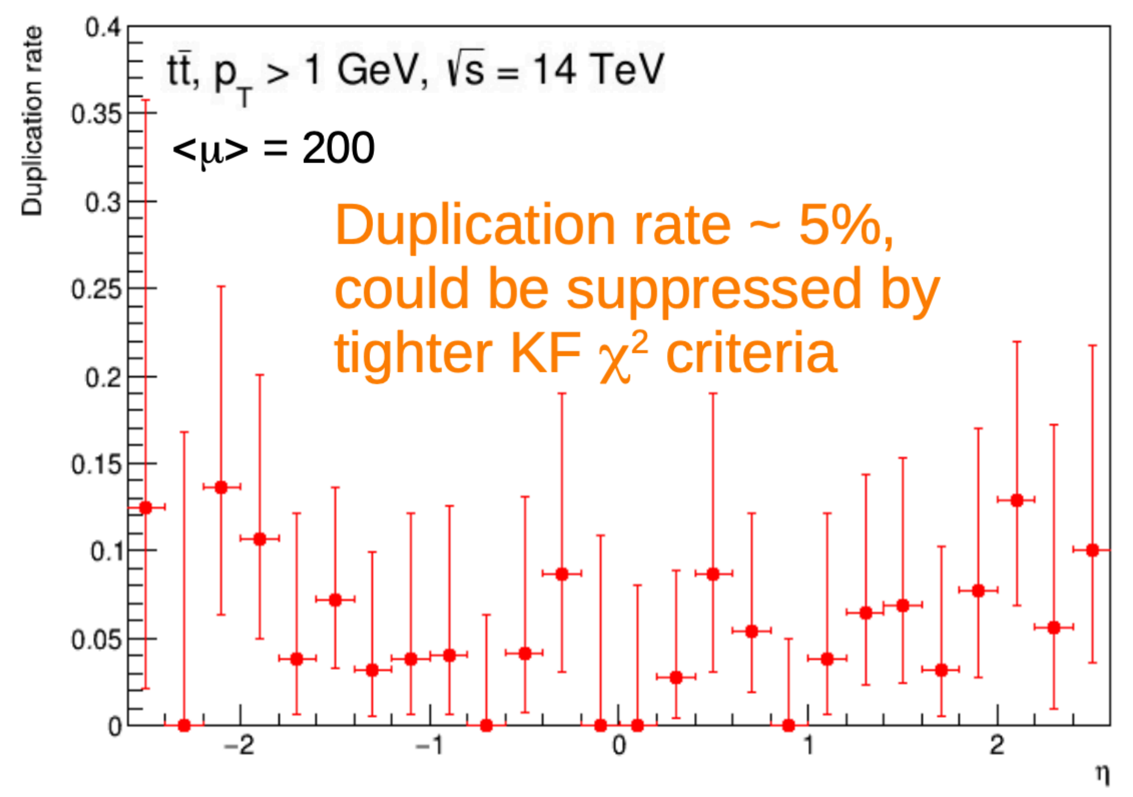
Efficiency vs. η



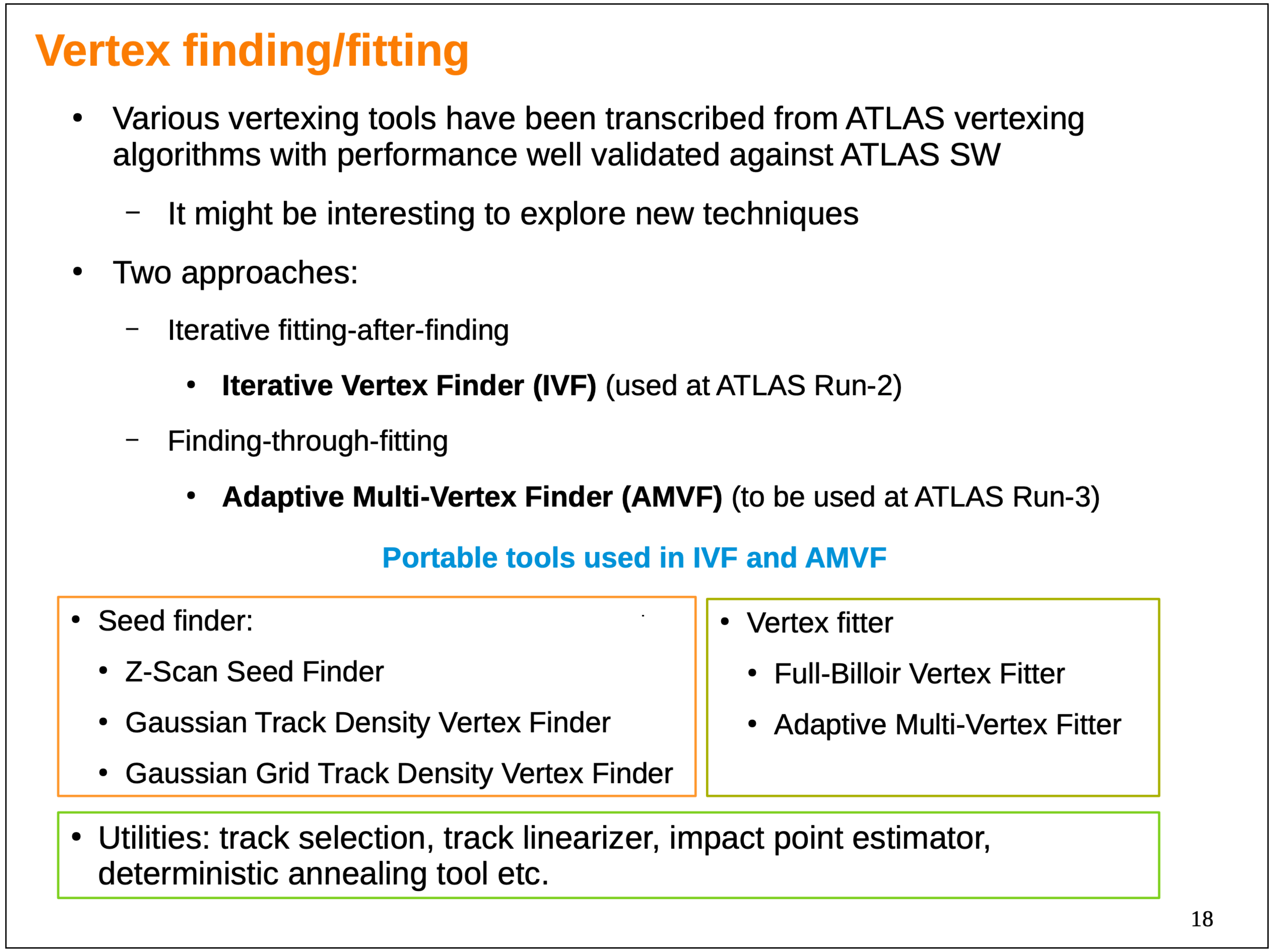
Fake rate vs. η



Duplication rate vs. η



Vertex Reconstruction



Vertex finding/fitting

- Various vertexing tools have been transcribed from ATLAS vertexing algorithms with performance well validated against ATLAS SW
  - It might be interesting to explore new techniques
- Two approaches:
  - Iterative fitting-after-finding
    - **Iterative Vertex Finder (IVF)** (used at ATLAS Run-2)
  - Finding-through-fitting
    - **Adaptive Multi-Vertex Finder (AMVF)** (to be used at ATLAS Run-3)

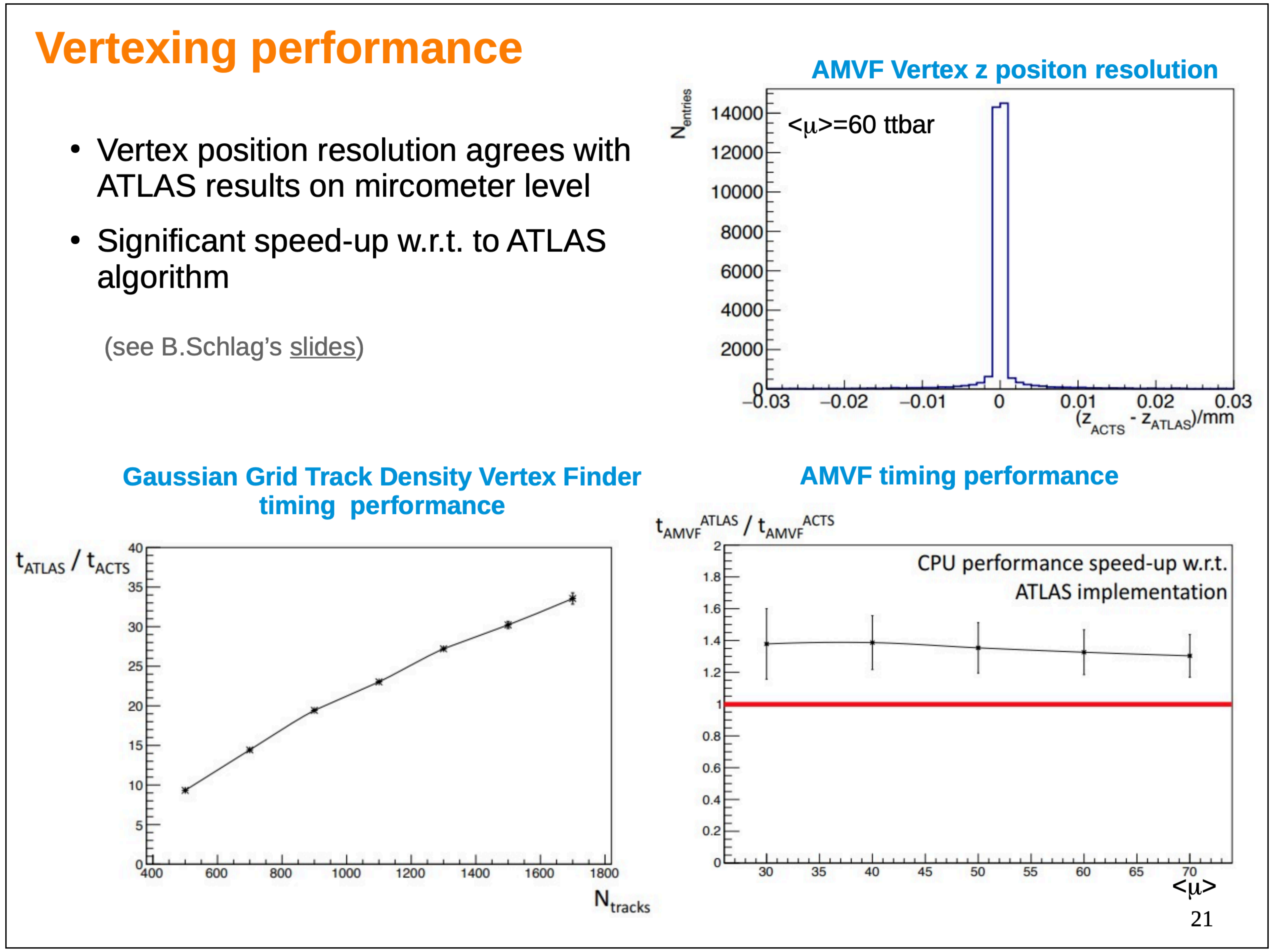
Portable tools used in IVF and AMVF

- Seed finder:
  - Z-Scan Seed Finder
  - Gaussian Track Density Vertex Finder
  - Gaussian Grid Track Density Vertex Finder

- Vertex fitter
  - Full-Billoir Vertex Fitter
  - Adaptive Multi-Vertex Fitter

- Utilities: track selection, track linearizer, impact point estimator, deterministic annealing tool etc.

18

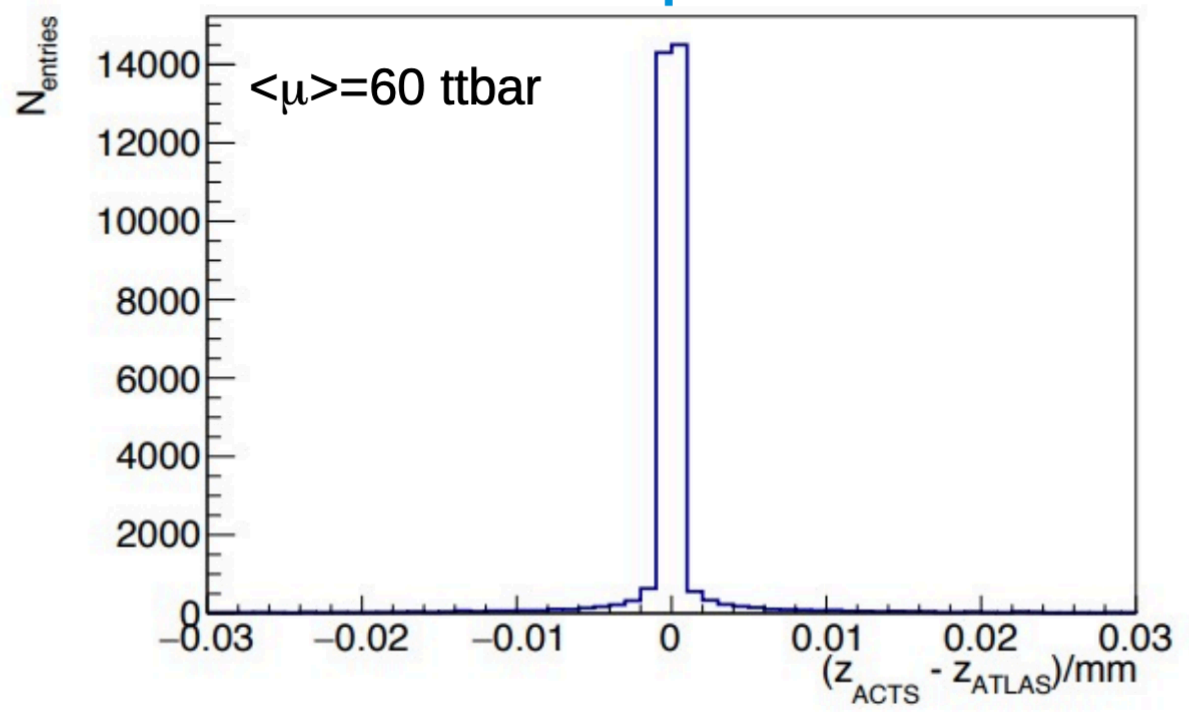


Vertexing performance

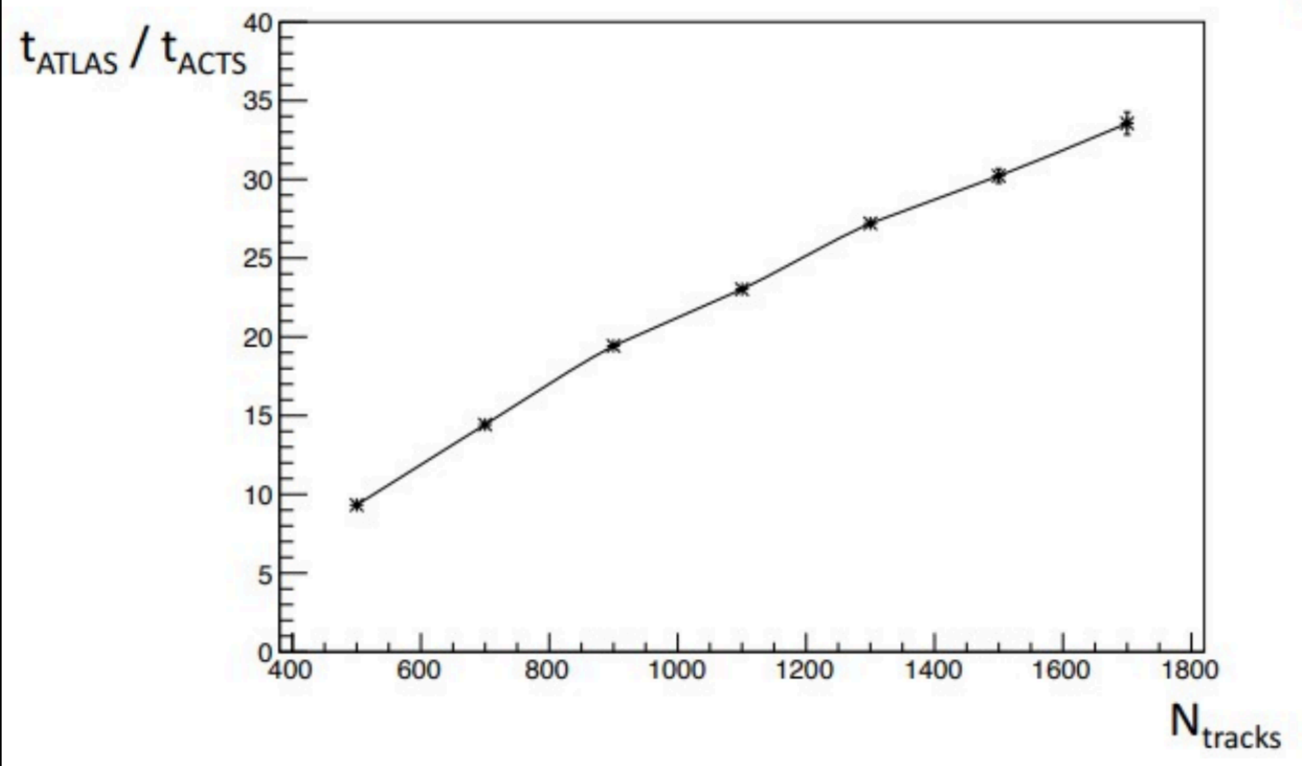
- Vertex position resolution agrees with ATLAS results on mircometer level
- Significant speed-up w.r.t. to ATLAS algorithm

(see B.Schlag's slides)

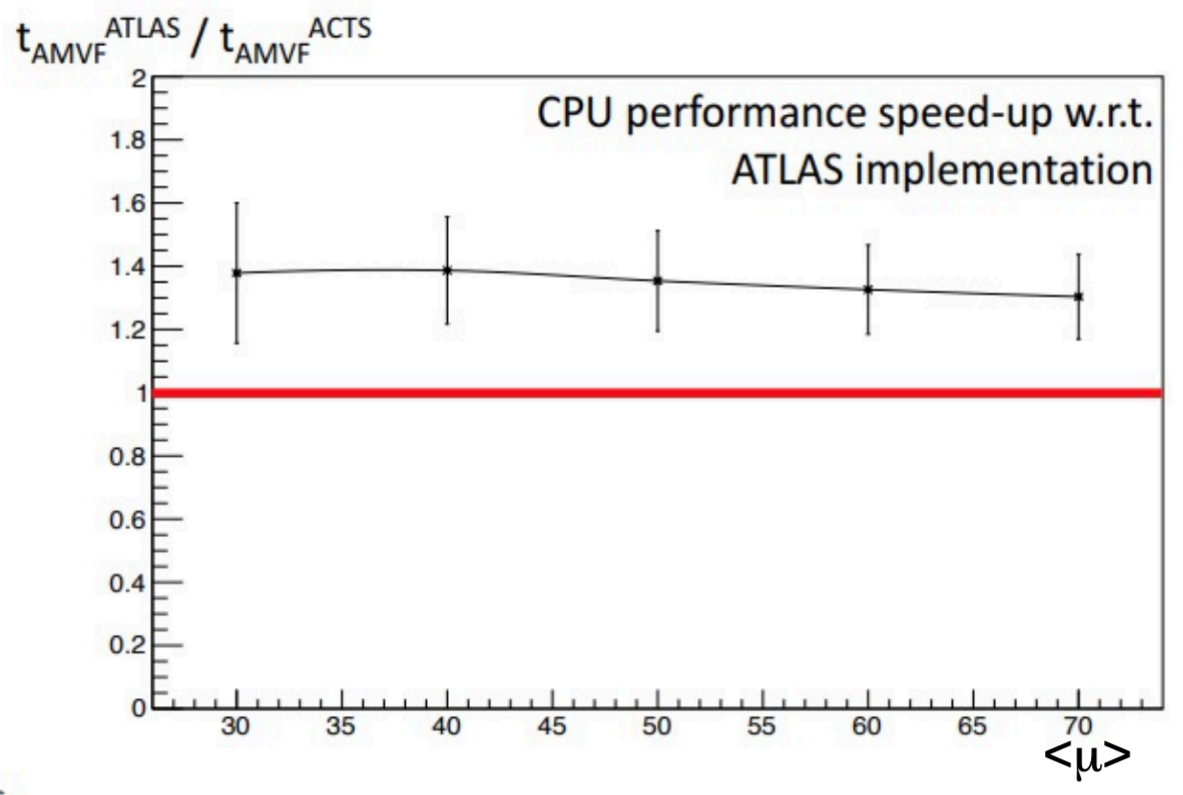
AMVF Vertex z positon resolution



Gaussian Grid Track Density Vertex Finder timing performance



AMVF timing performance



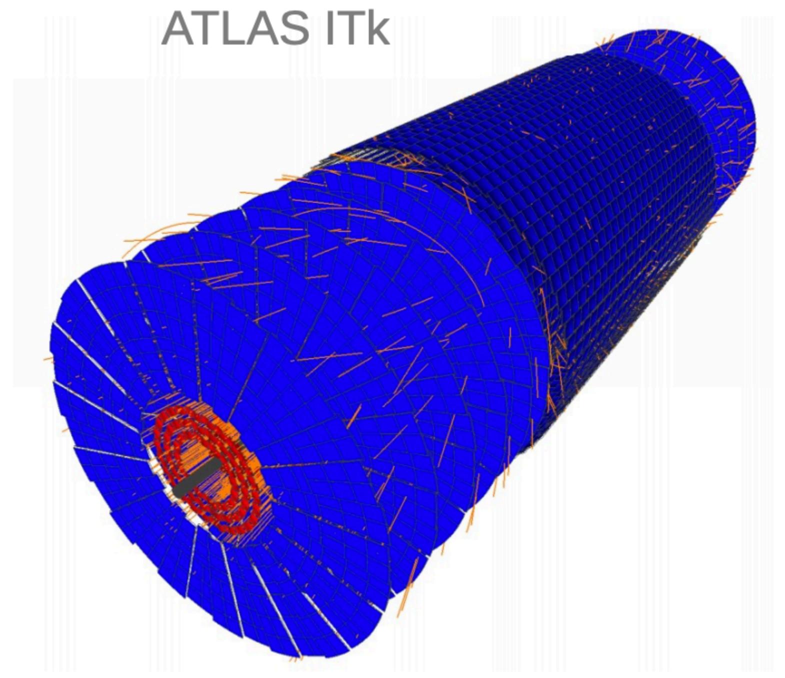
21

ACTS - Multi-experiment support

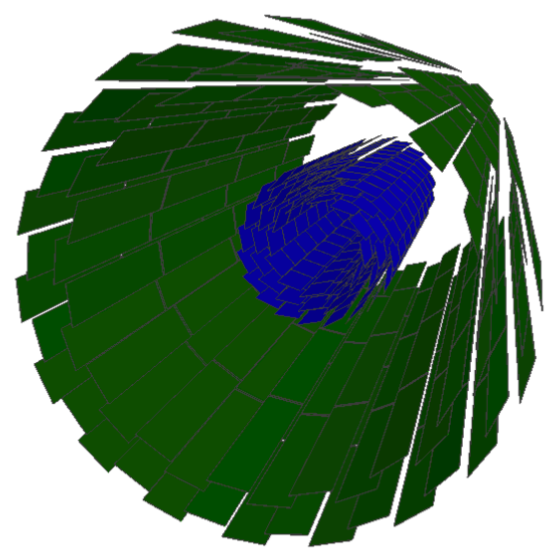
acts application to detectors

- Detector geometry implemented:
  - ATLAS ID+Calo, ATLAS ITK
  - Open Data Detector
  - FCC-hh
  - FASER Silicon
  - CEPC Silicon+TPC
  - sPHENIX Silicon
  - Belle-II Silicon
- On-going/planned implementation:
  - ATLAS Muon System
  - sPHENIX TPC (see J. Osborn's CTD talk)
  - Belle-II Drift Chamber

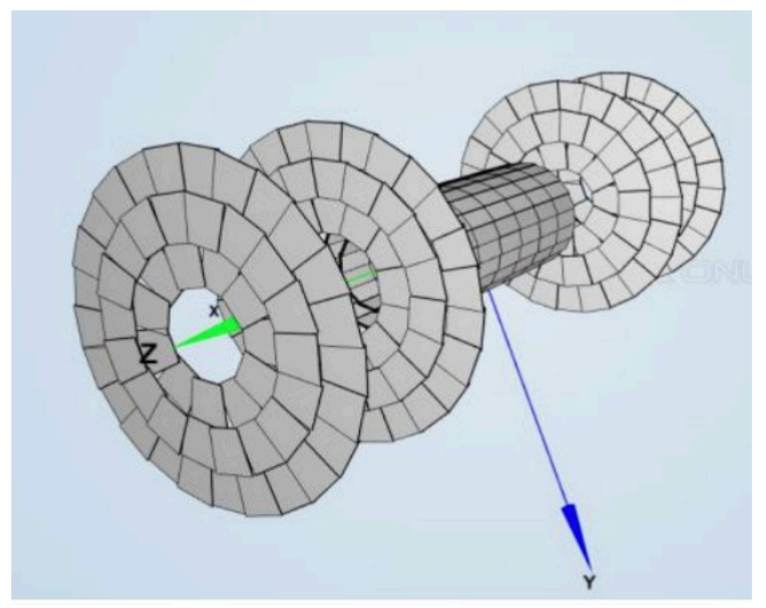
ATLAS ITk



sPHENIX Silicon



CEPC VTX

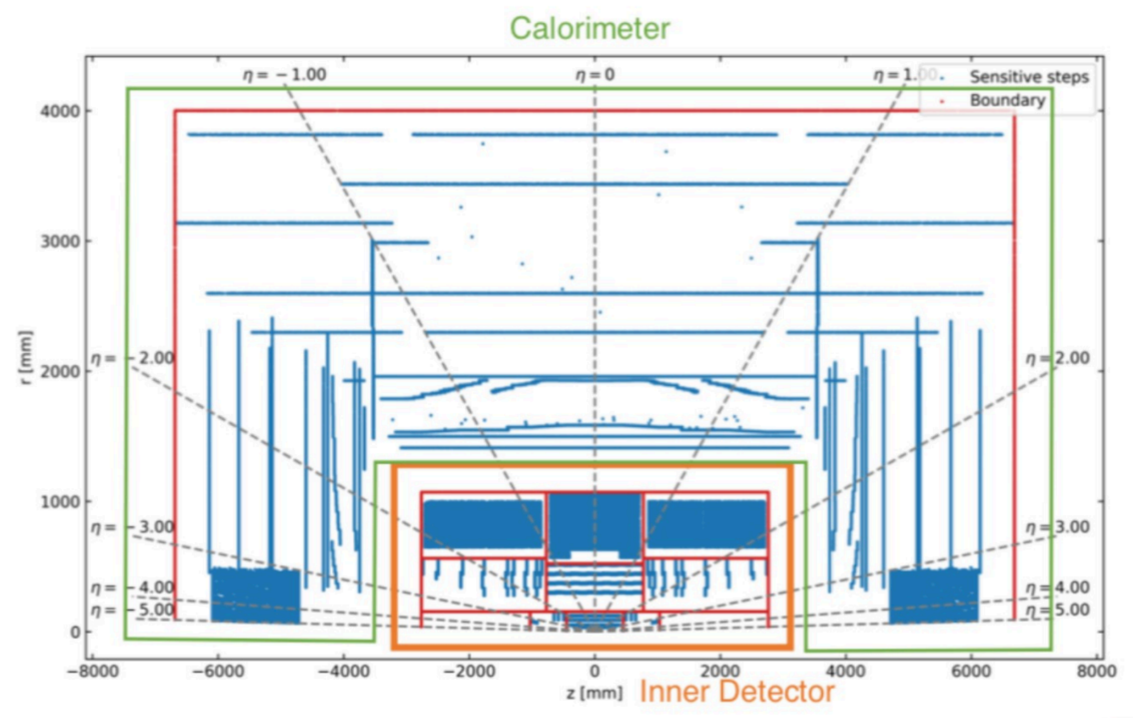


16

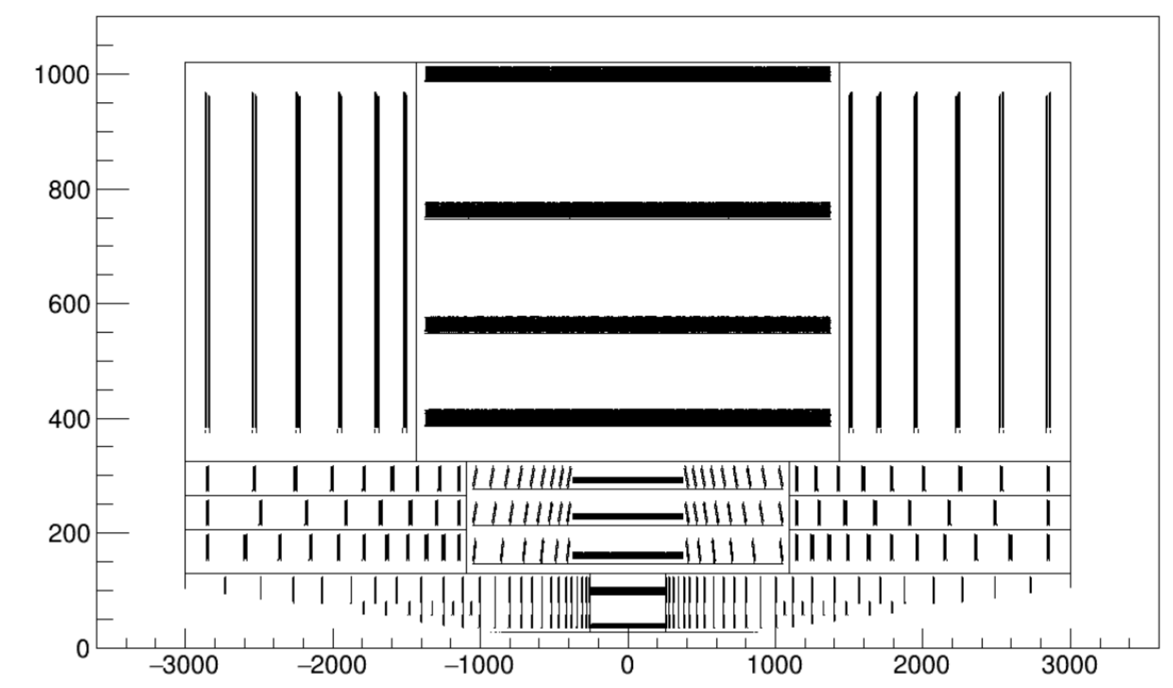
acts application to detectors

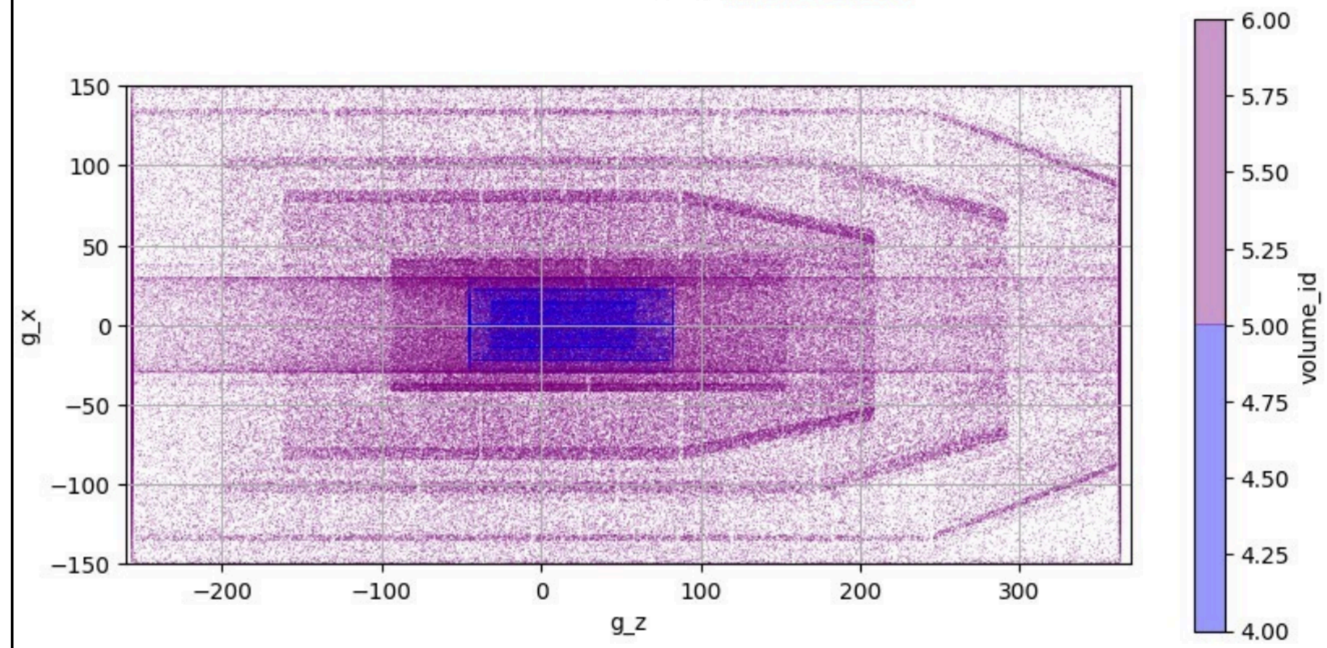
Propagation tests through implemented geometry

ATLAS ID + Calorimeter

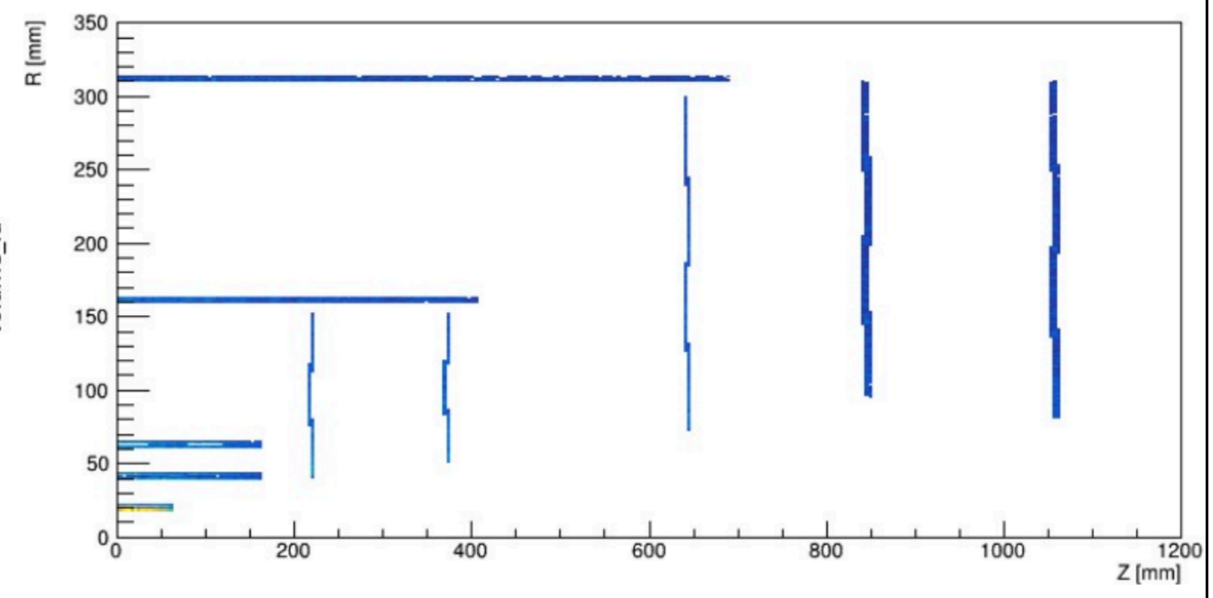


ATLAS ITk





Belle-II silicon



CEPC silicon

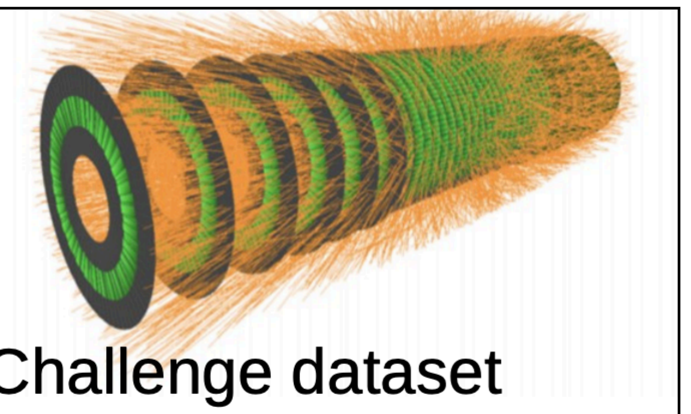
17

See detailed feedback from different experiments at the ACTS 2020 workshop

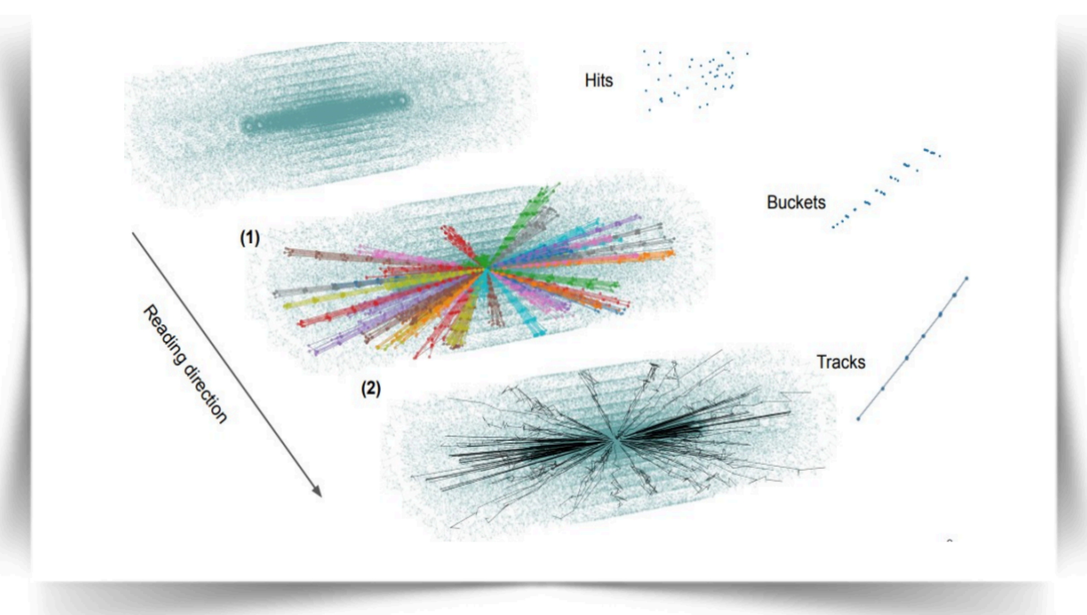
ACTS - R&D Platform

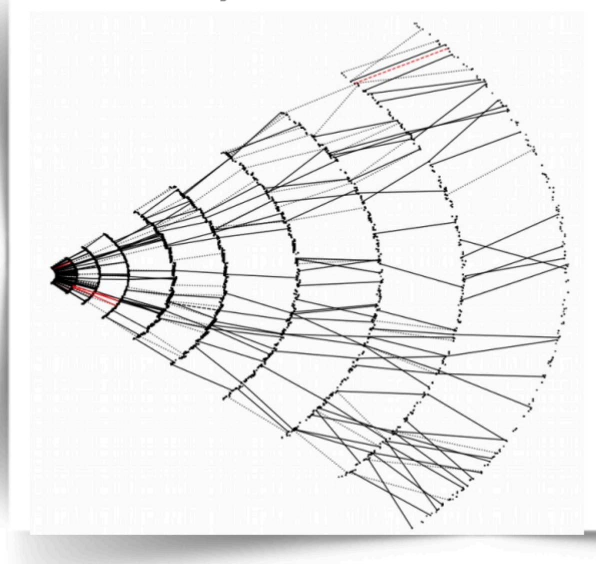
R&D with acts

Support for tracking techniques R&D



- ACTS fast simulation engine is used to generate TrackML Challenge dataset
- The TrackML dataset is used for various tracking algorithms R&D projects
  - Similarity Hashing and learning (see M. Kiehn's CTD talk)
  - Hep.TrkX & Exa.TrkX project (see D. Murnane's CTD talk)
  - The Mikado algorithm (see S Gorbunov's slides)





18

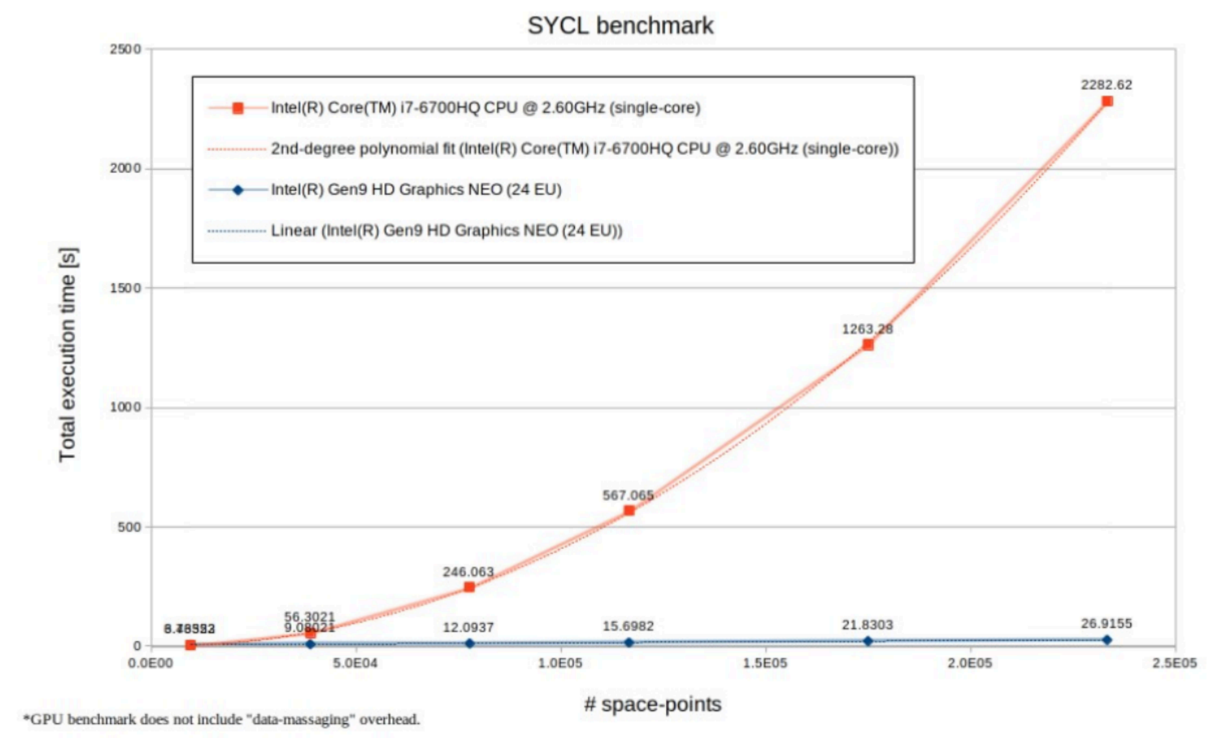
R&D with acts

GPUs-accelerated tracking

- Seed finder with SYCL/CUDA implementation on GPUs

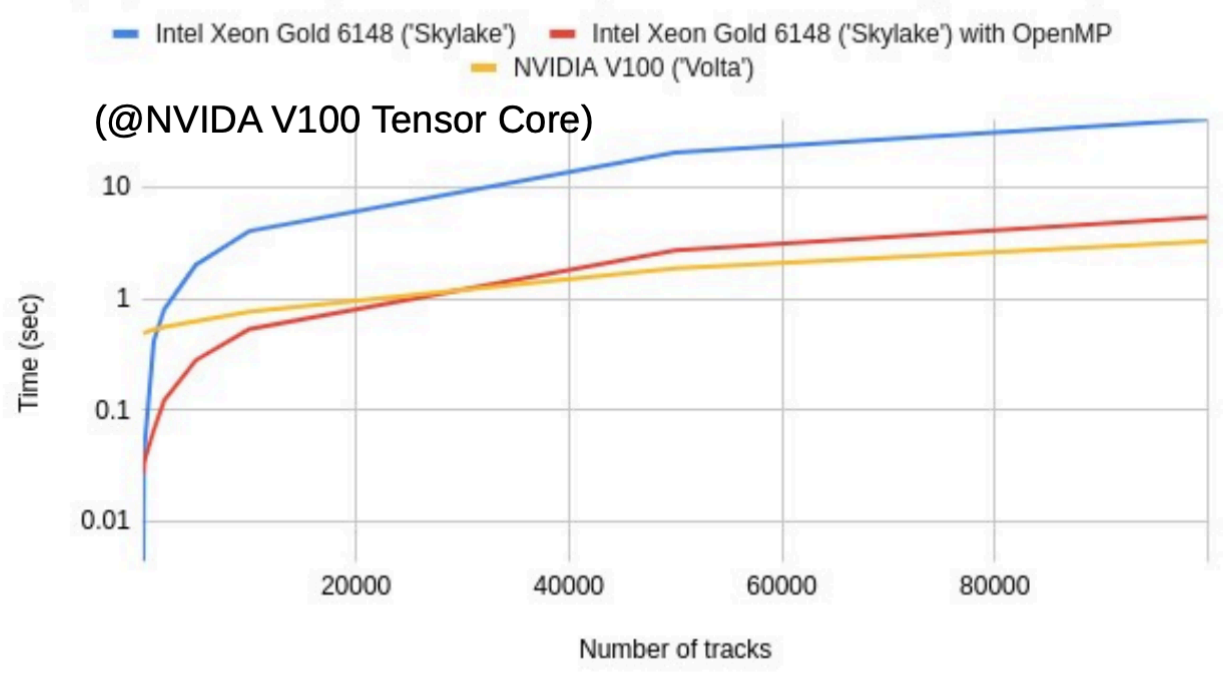
  (V. Pascuzzi, B. Yeo, C. Legett et. al)
- On-going efforts for a GPUs-based KalmanFilter (X. Ai et. al)
  - Free parameter stepping at ATLAS B field on GPUs with CUDA has been validated

ACTS SeedfinderTest (@Intel HD Graphics NE

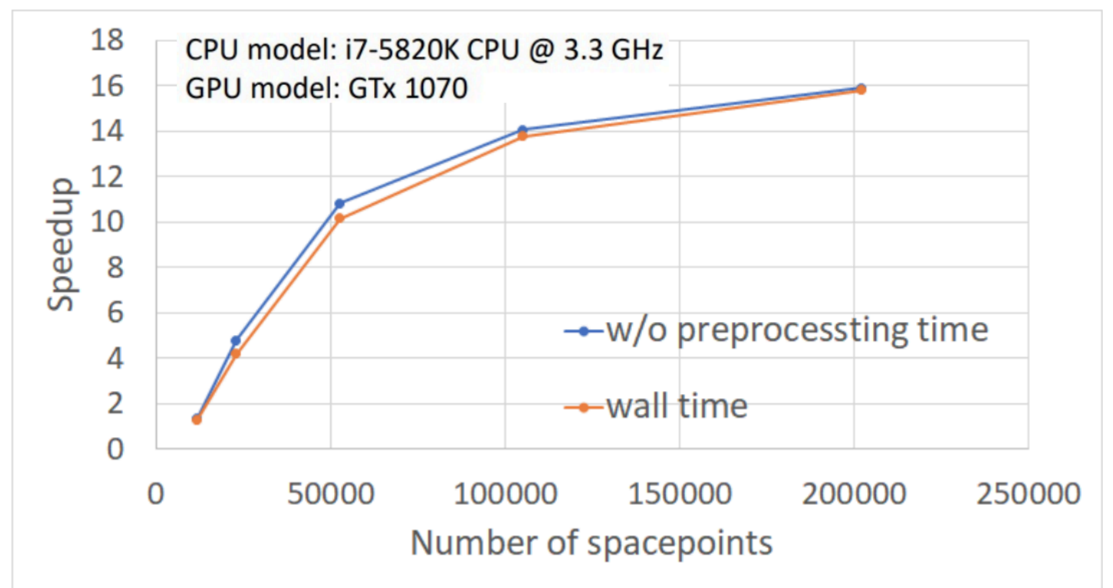


Propagation tests (ATLAS B Field, pT=0.1GeV/c, nSteps = 1000)

(@NVIDIA V100 Tensor Core)



ACTS seedfinder with CUDA (@NVIDIA GTX 1070)



20

See also: Ai, X., Mania, G., Gray, H.M. et al. A GPU-Based Kalman Filter for Track Fitting. Comput Softw Big Sci 5, 20 (2021)

Conclusions

- Shared tools can provide “ready-to-use” solutions for smaller experiments
  - providing access to core functionalities implemented in an experiment-agnostic way
  - experiment-specific aspects (geometry, material, calibrations, tuning, etc.) still require significant effort from each experiment
- They also provide a direct connection to R&D work!
  - both for new algorithms (AI) and computing platforms (HPC)