# Vectorized Geometry

git.cern.ch/pub/VecGeom

## A new generic C++ geometry library for detector-particle simulation

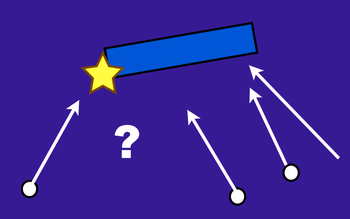
J. Apostolakis[1], A. Bhattacharyya[2], P. Canal[3], F. Carminati[1], G. Cosmo[1], J. De Fine Licht[1], A. Gheata[1], G. Lima[3], H. Kim[3], T. Nikitina[1], R. Sehgal[2], and S. Wenzel[1]
[1]CERN, [2]BARC, [3]Fermilab

## Motivation

- **Geometry is essential for particle-detector simulation;** navigating in complex geometries takes **~40%** of CPU time
- Prepare geometry for next generation simulation frameworks and future hardware
  - usage of SIMD instructions & accelerators ( GPU )

## Goals

- Optimized library of primitive solids & composites
- Optimized algorithms for track navigation
- APIs for single-track & many-tracks
- SIMD vectorization for many-tracks + optionally single-track
- GPU & accelerator ready
- Follow existing USolids [1] single-track API



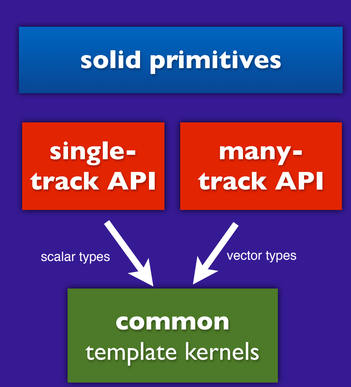

[1] M. Gayer *et al* 2012 *IOP Conf Proc.* **396** 052035

## Challenges

- **Achieve good vector & GPU performance**
- **Avoid code duplication** ( from new API and platforms )

## Methods

- Core of **templated code kernels** using abstract types
- Kernels used to implement single-track, many-track and GPU methods
- Use **vector types** (Vc [2]) for portable and reliably performant vectorization
- Compile time branch elimination using shape specialization (e.g., full vs hollow tube)
- Placed-solid classes combine transformation & solid types




[2] M. Kretz and V. Lindenstruth, „Vc: A C++ library for explicit vectorization", Software: Practice and Experience, 2011.

## Implementation Status

- Implementations of all solids in **CMS 2015 detector**
  - generic kernels for Box, Tube, Trapezoid, Polyhedra, Torus, Trd, Boolean Solids
  - simple implementation for Cone, Polycone -- without vectorization
- First version of single & many-track navigation
  - can trace in complex detectors such as CMS
  - without voxelization
- VecGeom compiles for CUDA devices and Intel Xeon Phi

## Testing / Reliability

- Adapted & extended test suites of Geant4, ROOT & USolids
- Created large database of test cases
- Cross comparisons with Geant4 & ROOT for single solid & navigation functionality
  - also identified issues in Geant4, ROOT implementations
- Generic benchmarking tools enable quick performance assessment (see Fig. 1)

## VecGeom Shape Performance Example

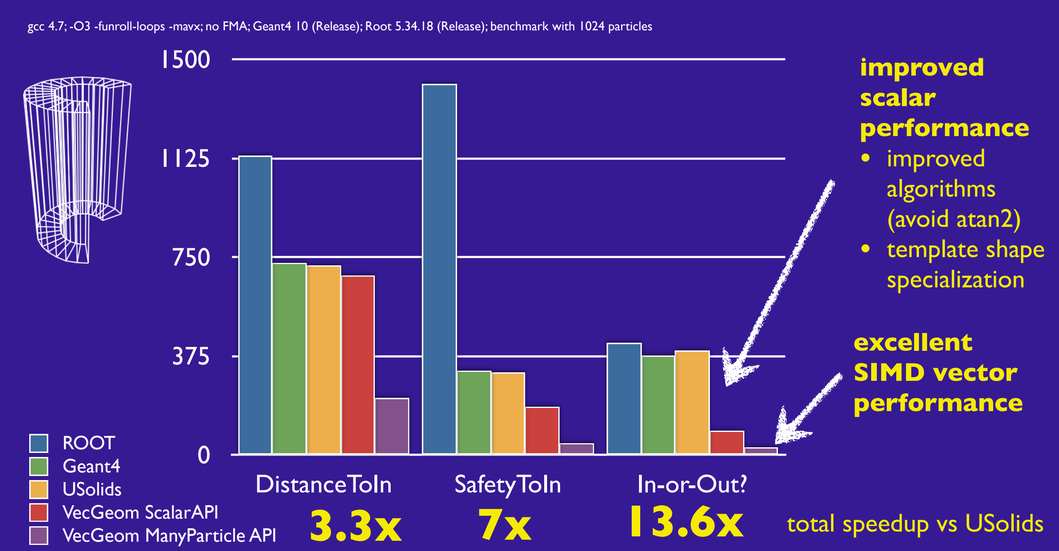
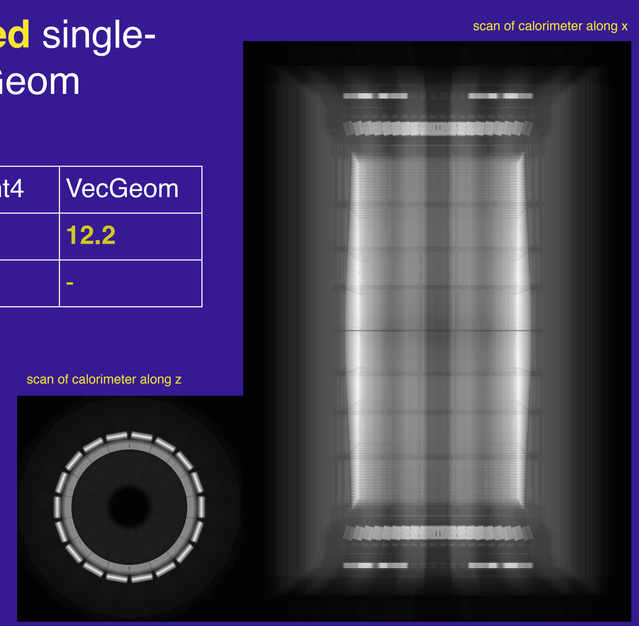


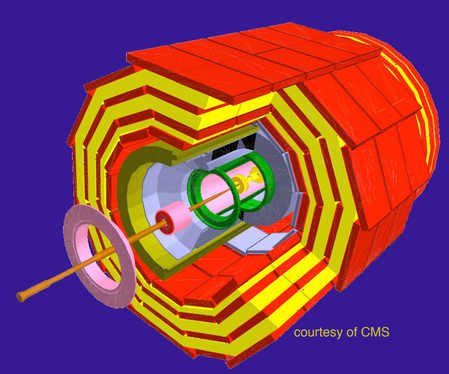

Fig. 1: Performance comparison of the tube-segment shape

## Global Performance Comparison

- „X-Ray" scan benchmark with CMS calorimeter
  - pixel-by-pixel tracing of geantino rays traversing setup
  - stresses all aspects of geometry package
    - performance & validation per track-level possible
    - validated against Geant4, ROOT using image & total number of boundary crossings
- Indicates **globally improved** single-track performance of VecGeom (e.g., runtime in s for scan along x)

| Voxelization | ROOT | Geant4 | VecGeom |
|---|---|---|---|
| OFF | 31.5 | 47.5 | 12.2 |
| ON | 11.7 | 19.2 | - |




## Testing the Vector API

- Compared tracing single tracks to tracing multiple identical tracks using the vector API ( for diverse geantino rays starting at CMS origin )
  - Validating vector API
  - Preliminary global **vector speedup of ~2.1** observed ( Intel iCore7 + SSE )

## Plans

- Vectorized kernel implementation of additonal shapes (Polycone, Extruded solid)
- Optimization of navigation using vectorizable voxel techniques
- Improvement of robustness in preparation for routine usage in Geant-V & Geant4
- Consolidate combined development with USolids library

## Conclusion

- **Completed set of solids for realistic detector geometry**
  - efficient vectorization for most shapes
- **First implementation of navigation**
- **Promising performance (in complex detectors) comparing to Geant4 & ROOT**
  - even in single-track mode
  - first vectorization speedup in many-track mode