VecGeom – Geometry for GeantV, and …

Sandro Wenzel (CERN), presented by Philippe Canal (FNAL)

for the VecGeom developers

G.Amadio (UNESP), J.Apostolakis (CERN), M.Bandieramonte* (CERN), Ph.Canal (FNAL), F. Carminati (CERN), G.Cosmo (CERN), J.DeFine Licht* (CERN), A.Gheata (CERN), M.Gheata (CERN), G.Lima (FNAL), T.Nikitina (CERN), R.Sehgal (BARC), S.Wenzel (CERN), Y.Zhang* (KIT)

* previously active

Outline

- **VecGeom – An Introduction**
  - Motivations; Requirements; Development Approach; Components
- **Shape Primitives — Status + Performance**
  - Feature overview
  - Implementation status
  - Reasons for speedup (highlights)
- **Navigation Module — Status + Performance**
  - Overview of developments
  - SIMD accelerated voxel structures
  - Navigator code generation
- **Summary, Future Plans**

VecGeom: Requirements + Goals

- **Geometry is one of the most important pillars of simulation**
- **GeantV needs geometry library …**

VecGeom: Requirements + Goals

- **Geometry is one of the most important pillars of simulation**
- **GeantV needs geometry library …**
  - able to handle baskets/vectors of tracks in all algorithmic parts (basket API)

VecGeom: Requirements + Goals

- **Geometry is one of the most important pillars of simulation**
- **GeantV needs geometry library …**
  - able to handle baskets/vectors of tracks in all algorithmic parts (basket API)
  - designed to be used in heavily multi-threaded environments + allowing for rapid track (context) switches

VecGeom: Requirements + Goals

- **Geometry is one of the most important pillars of simulation**
- **GeantV needs geometry library …**
  - able to handle baskets/vectors of tracks in all algorithmic parts (basket API)
  - designed to be used in heavily multi-threaded environments + allowing for rapid track (context) switches
  - able to compile/run on accelerators (GPU, KNL, …)

VecGeom: Requirements + Goals

- **Geometry is one of the most important pillars of simulation**
- **GeantV needs geometry library …**
  - able to handle baskets/vectors of tracks in all algorithmic parts (basket API)
  - designed to be used in heavily multi-threaded environments + allowing for rapid track (context) switches
  - able to compile/run on accelerators (GPU, KNL, …)
- **Target performance and SIMD acceleration in all aspects**

VecGeom: Requirements + Goals

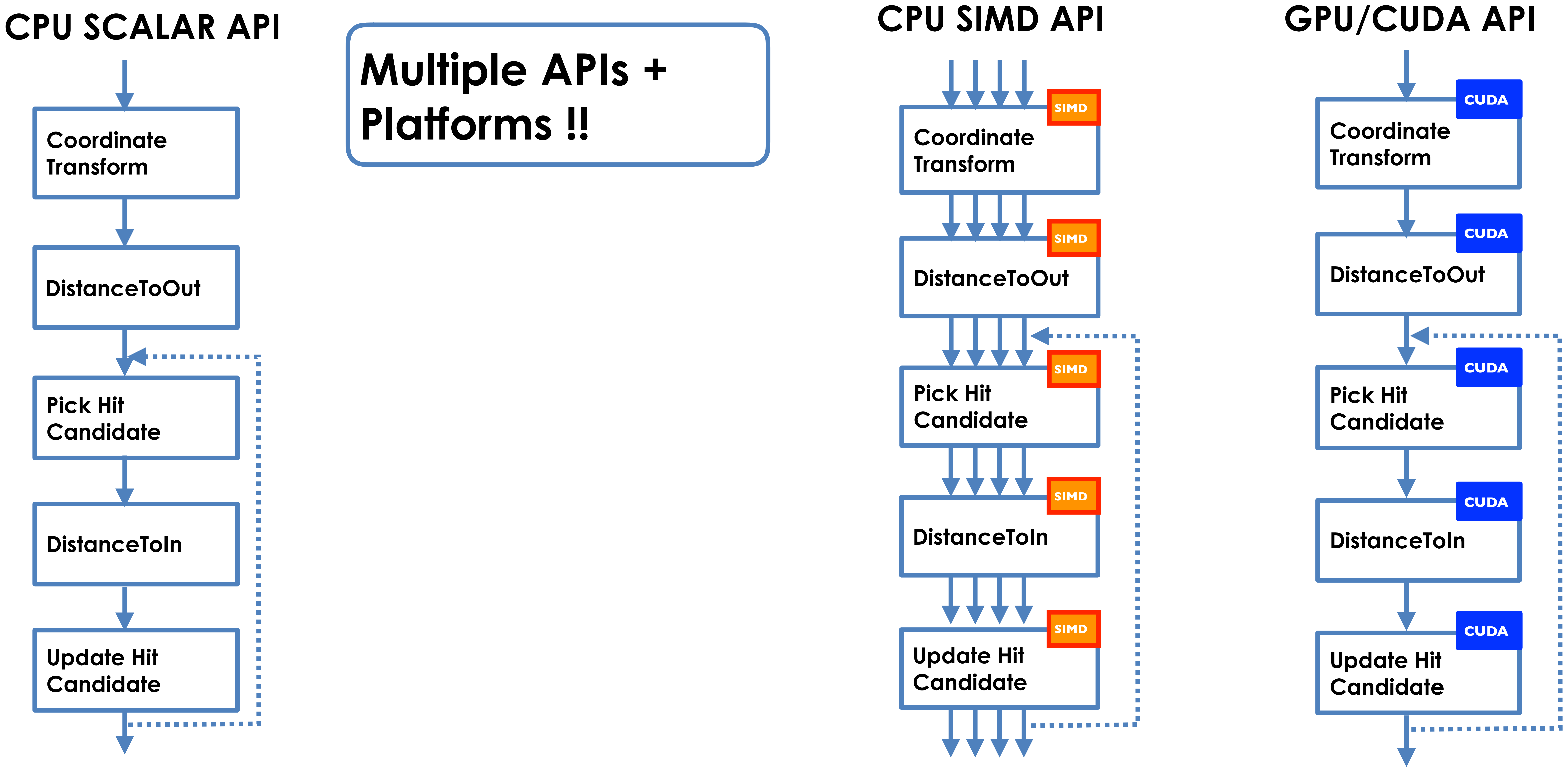
- **Geometry is one of the most important pillars of simulation**
- **GeantV needs geometry library …**
  - able to handle baskets/vectors of tracks in all algorithmic parts (basket API)
  - designed to be used in heavily multi-threaded environments + allowing for rapid track (context) switches
  - able to compile/run on accelerators (GPU, KNL, …)
- **Target performance and SIMD acceleration in all aspects**
  - Target SIMD speedup by handling vectors of tracks

VecGeom: Requirements + Goals

- **Geometry is one of the most important pillars of simulation**
- **GeantV needs geometry library …**
  - able to handle baskets/vectors of tracks in all algorithmic parts (basket API)
  - designed to be used in heavily multi-threaded environments + allowing for rapid track (context) switches
  - able to compile/run on accelerators (GPU, KNL, …)
- **Target performance and SIMD acceleration in all aspects**
  - Target SIMD speedup by handling vectors of tracks
  - Or SIMD speedup of complex algorithm handling one track (internal vectorization)

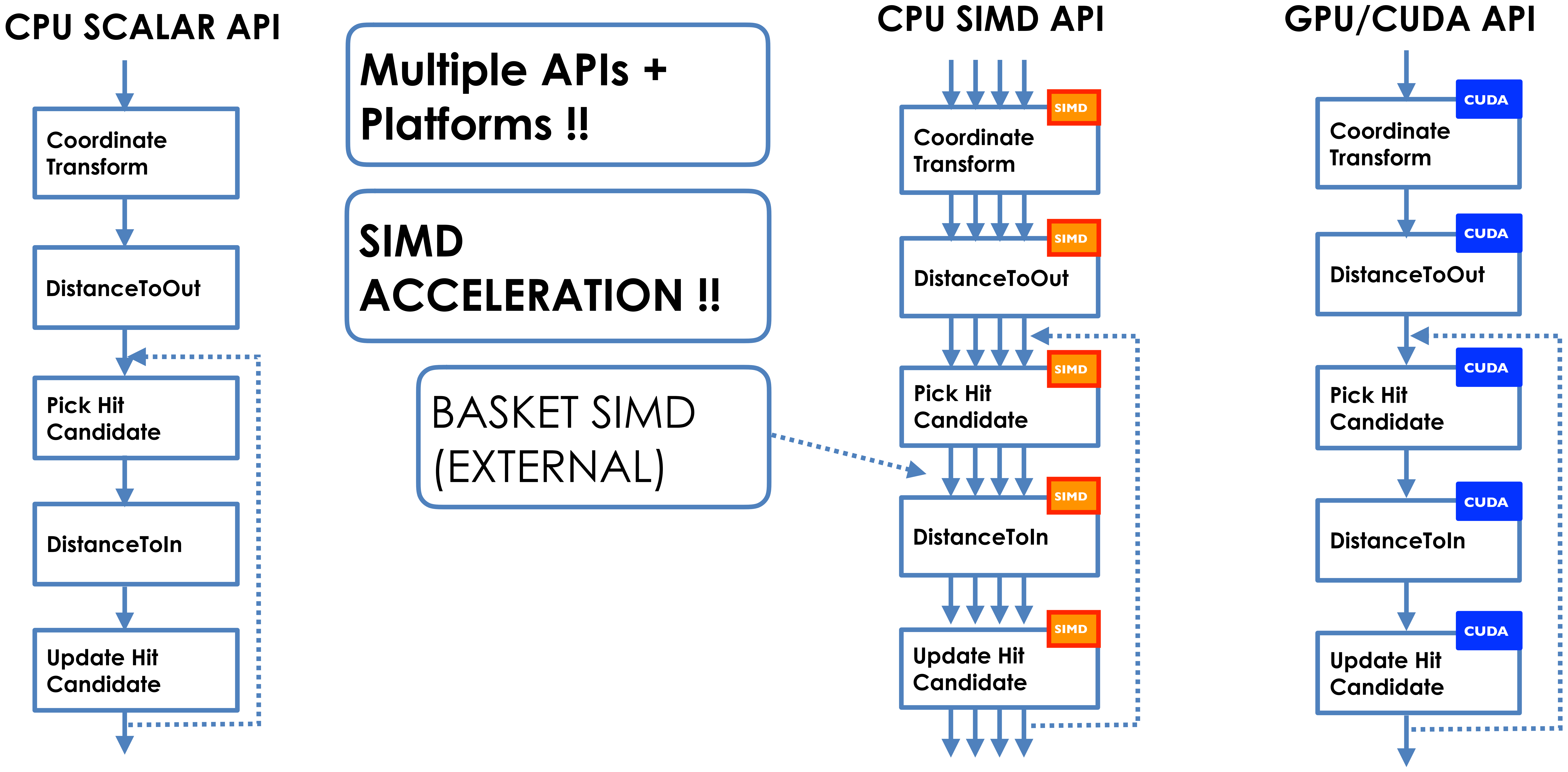
Motivations illustrated …

Multiple APIs + Platforms !!



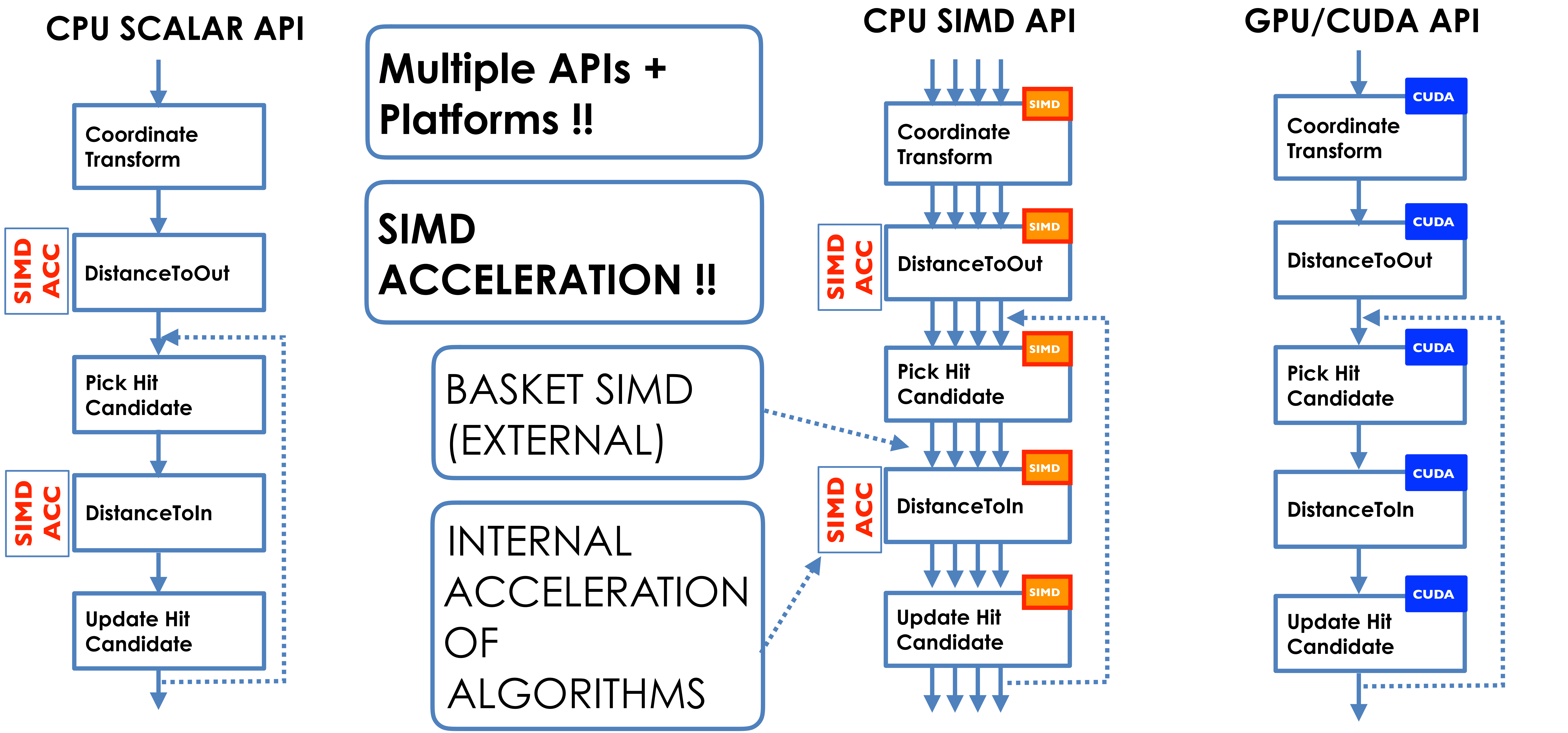
from typical algorithmic pipeline in navigation

Motivations illustrated …



from typical algorithmic pipeline in navigation

Motivations illustrated …



from typical algorithmic pipeline in navigation

VecGeom: Challenges

- **Substantial code management / duplication problem**
  - Support both traditional “scalar” track queries as well as “vector” queries
  - Need to compile on a lot more platforms than previously

VecGeom: Challenges

- **Substantial code management / duplication problem**
  - Support both traditional “scalar” track queries as well as “vector” queries
  - Need to compile on a lot more platforms than previously
- **How to SIMD-vectorize reliably?**
  - How to vectorize for basket case?
  - How to internally accelerate complicated shape primitives or scenes?

Strategy To Address Challenges

- **Promote use of components of generic templated code…**
  - to describe the algorithms once; instantiate them for different scenarios (scalar; vector; GPU) to solve code duplication
  - for performance via compile time code specialization

Strategy To Address Challenges

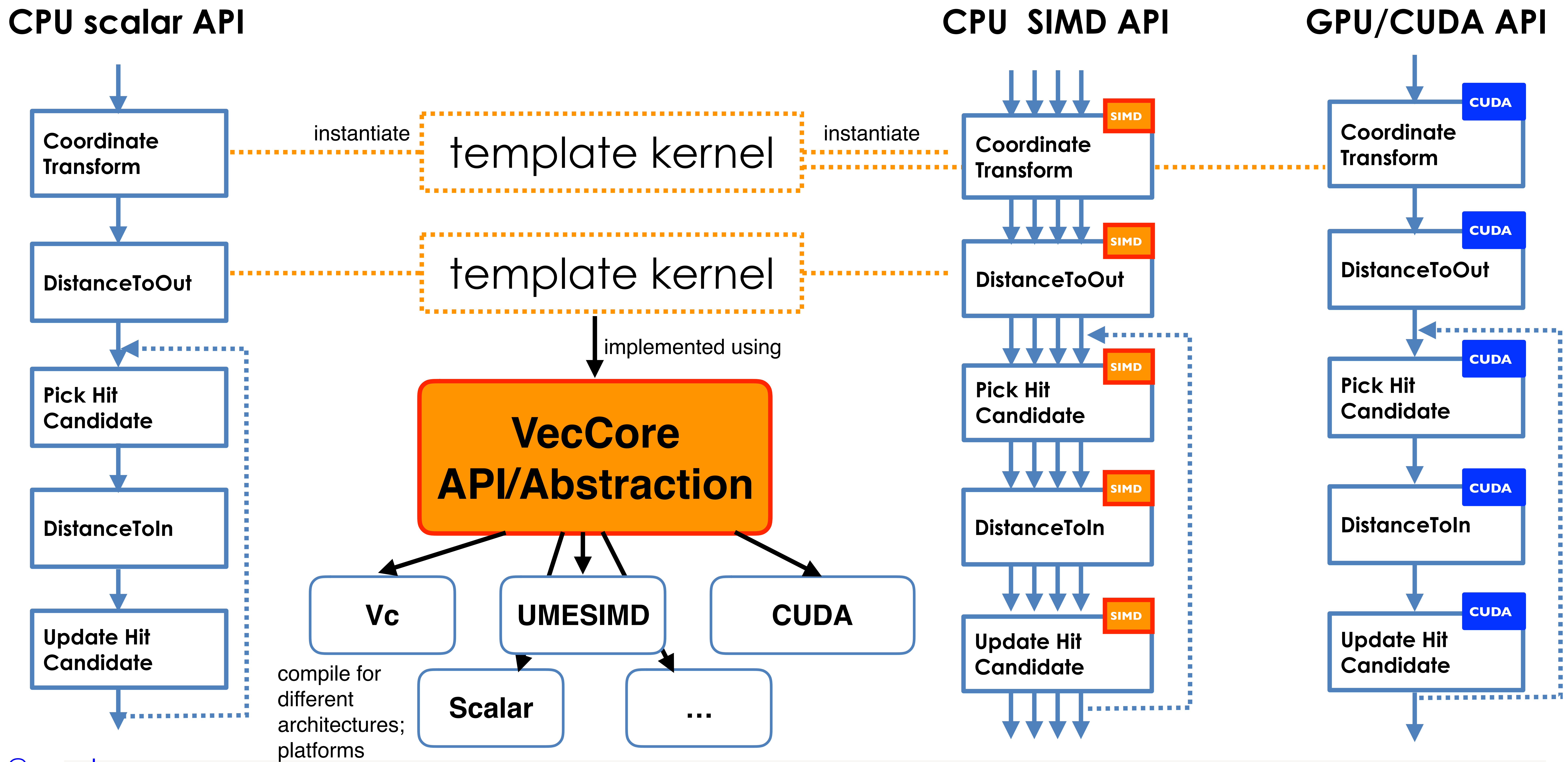
- **Promote use of components of generic templated code…**
  - to describe the algorithms once; instantiate them for different scenarios (scalar; vector; GPU) to solve code duplication
  - for performance via compile time code specialization
- **Use explicit vectorization techniques through SIMD wrapper libraries and VecCore abstraction**

Strategy To Address Challenges

- **Promote use of components of generic templated code…**
  - to describe the algorithms once; instantiate them for different scenarios (scalar; vector; GPU) to solve code duplication
  - for performance via compile time code specialization
- **Use explicit vectorization techniques through SIMD wrapper libraries and VecCore abstraction**
- Start from existing code from G4 / TGeo / USolids but review/ improve/redesign algorithms and memory layout

Generic Programming Approach Illustrated

The architectural pattern pioneered in VecGeom



VecGeom: Component Overview

VecGeom

Geometry Primitives (USolids)

- Box
- Tube
- Cone
- ...

Navigation Module

- Navigators
- NavigationState

Geometry Modeller To Build Hierachical Detectors

- LogicalVolume
- PlacedVolume
- Transformation
- ...

Scalar (CPU + GPU) APIs

Multi-Track (CPU) SIMD APIs

VecGeom and USolids

- **USolids originated** as a project before VecGeom with the goal to unify "shape-primitives" development for **Geant4/TGeo** (AIDA financed project)

- Today, VecGeom contains the USolid effort and is the natural evolution of it

- **USolids now is** the geometry-primitive part of VecGeom, targeted to unify "shape-primitive" development for **Geant4/TGeo/Geant-V**

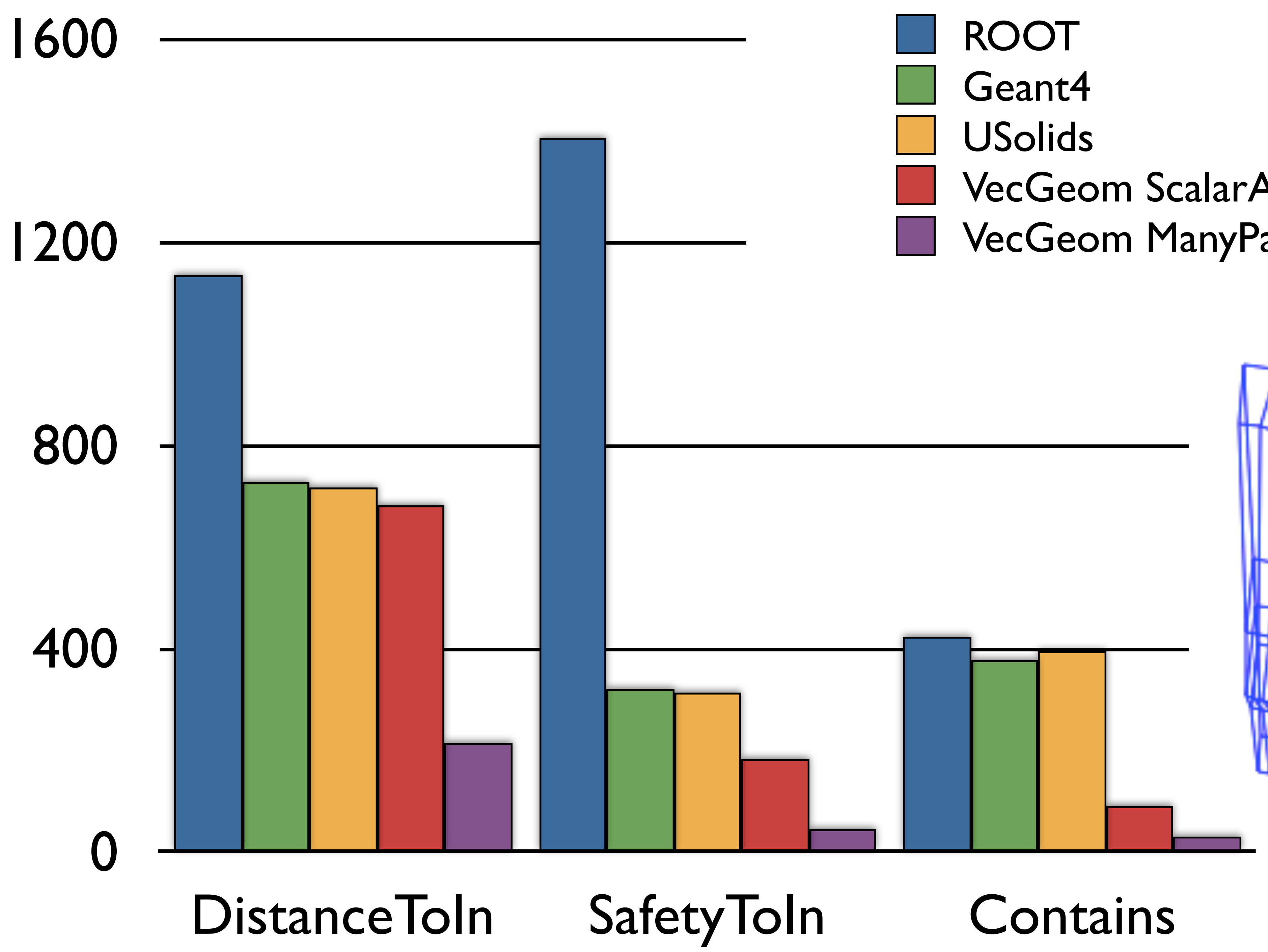
- Many of the good results today are due to the original USolids effort

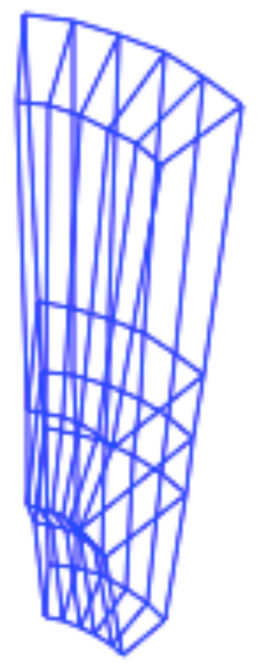
Geometry Primitives

Shape Primitives - Highlights

- **Better algorithms** compared to previous implementations

- **Basket interface** with SIMD gains (in simple geometry primitives)





Time (units) measured for a tube segment

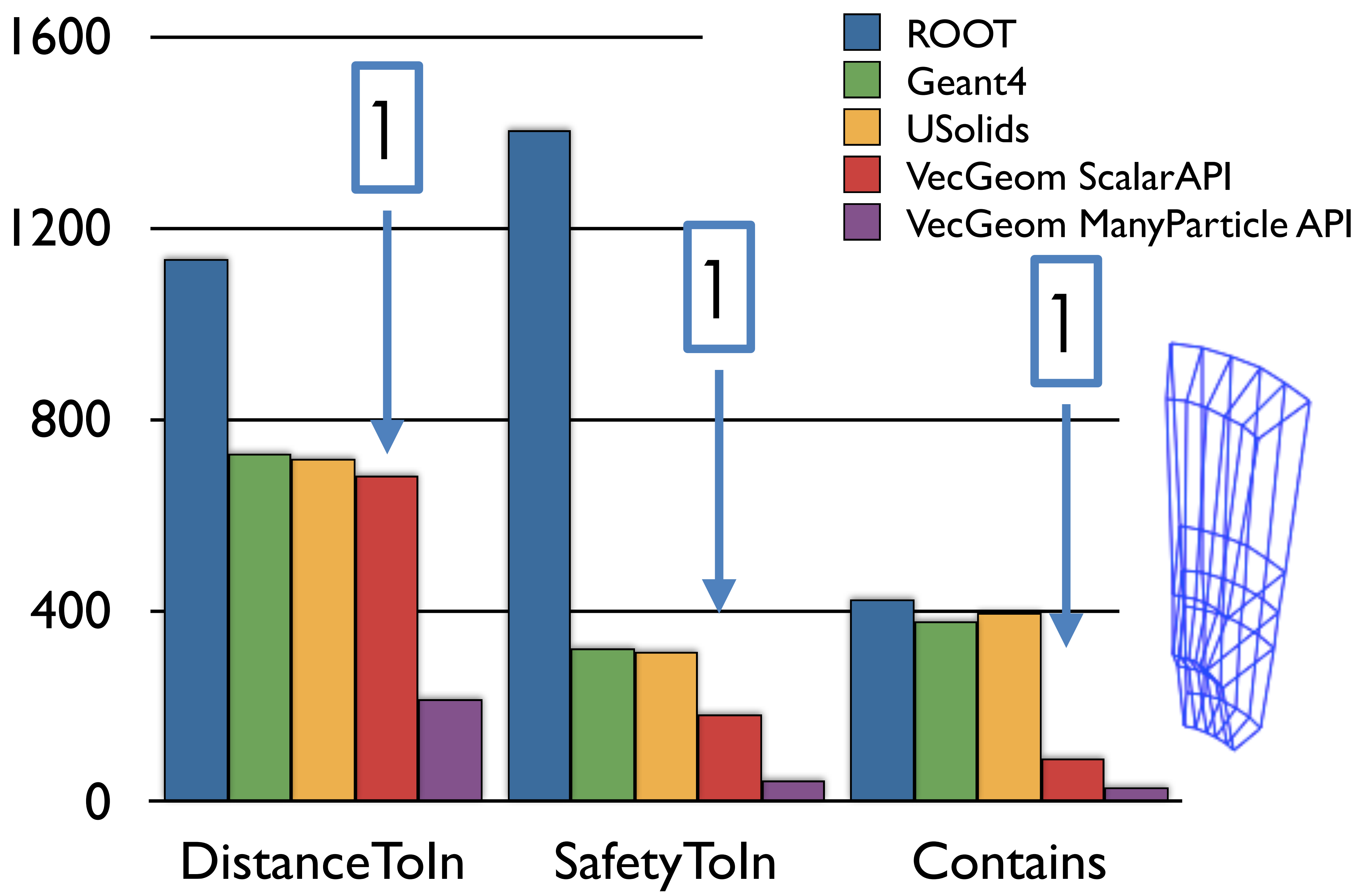
NEEDS UPDATE

Shape Primitives - Highlights

- **Better algorithms** compared to previous implementations

1

- **Basket interface** with SIMD gains (in simple geometry primitives)

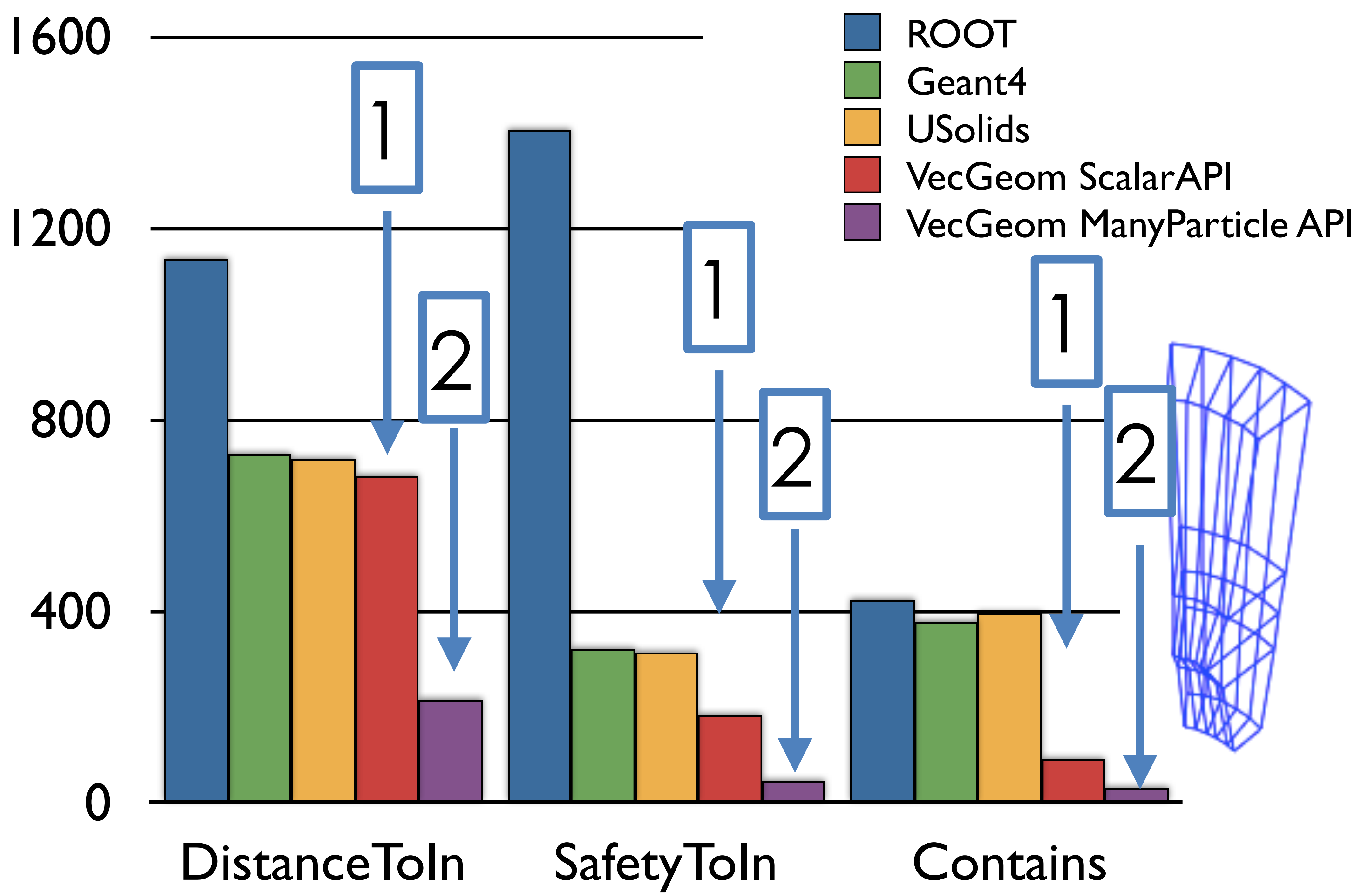


Time (units) measured for a tube segment

NEEDS UPDATE

Shape Primitives - Highlights

- **Better algorithms** compared to previous implementations 1
- **Basket interface** with SIMD gains (in simple geometry primitives) 2

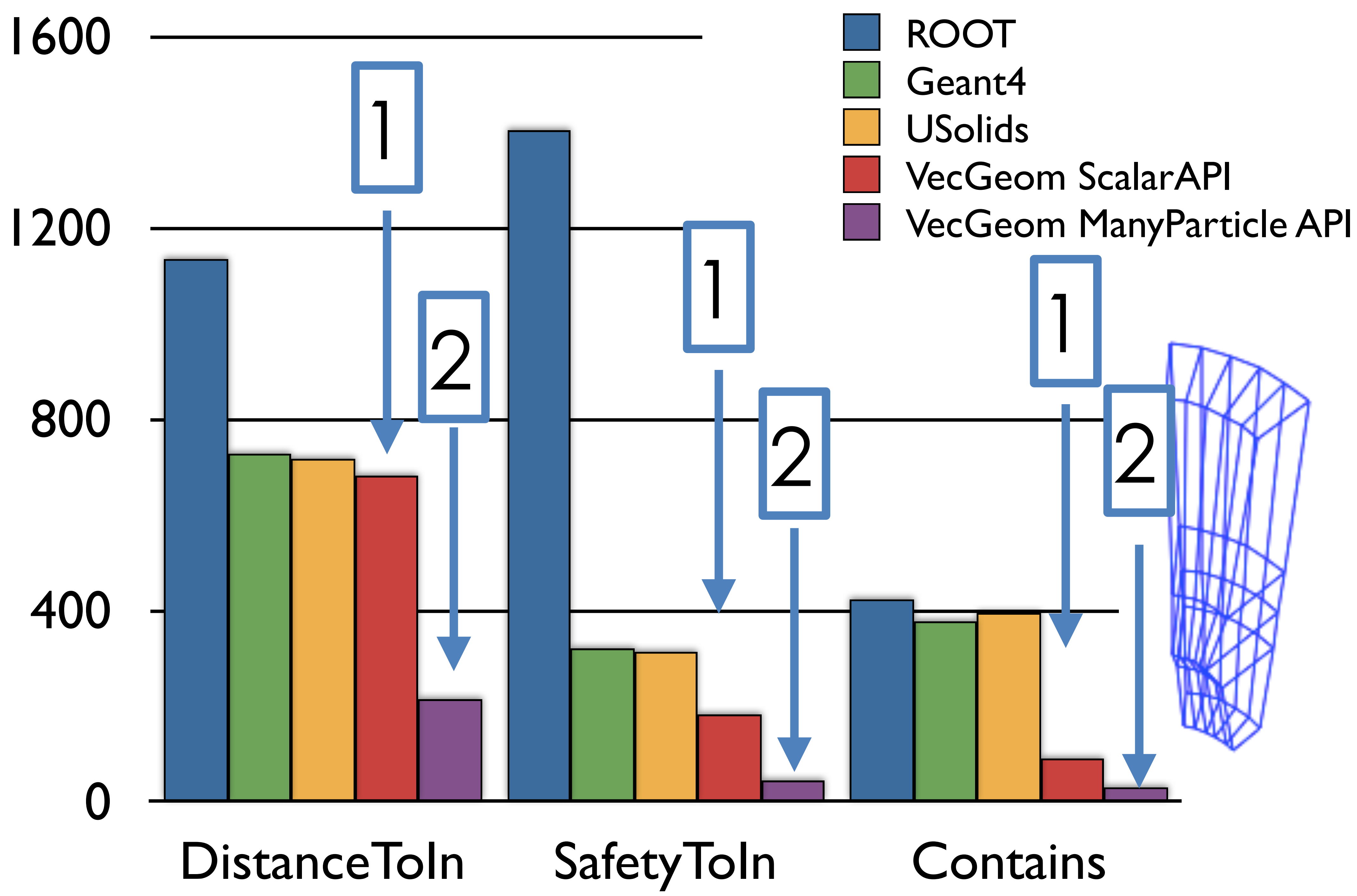


Time (units) measured for a tube segment

NEEDS UPDATE

Shape Primitives - Highlights

- **Better algorithms** compared to previous implementations [1]
- **Basket interface** with SIMD gains (in simple geometry primitives) [2]



Time (units) measured for a tube segment

- **Portable code** (CPU and GPU/CUDA)
- **Portable SIMD** (SSE to AVX-512)

Speedup of basket treatment (100000 tracks) against VecGeom scalar CPU version

	SSE	AVX2	AVX-512	CUDA
DistanceToIn	1.57	2.07	2.51	tbd
Safety	2.37	2.50	3.56	tbd
Contains	1.72	2.17	3.36	tbd

For details on AVX-512 and CUDA, see talk on “Accelerators”

NEEDS UPDATE

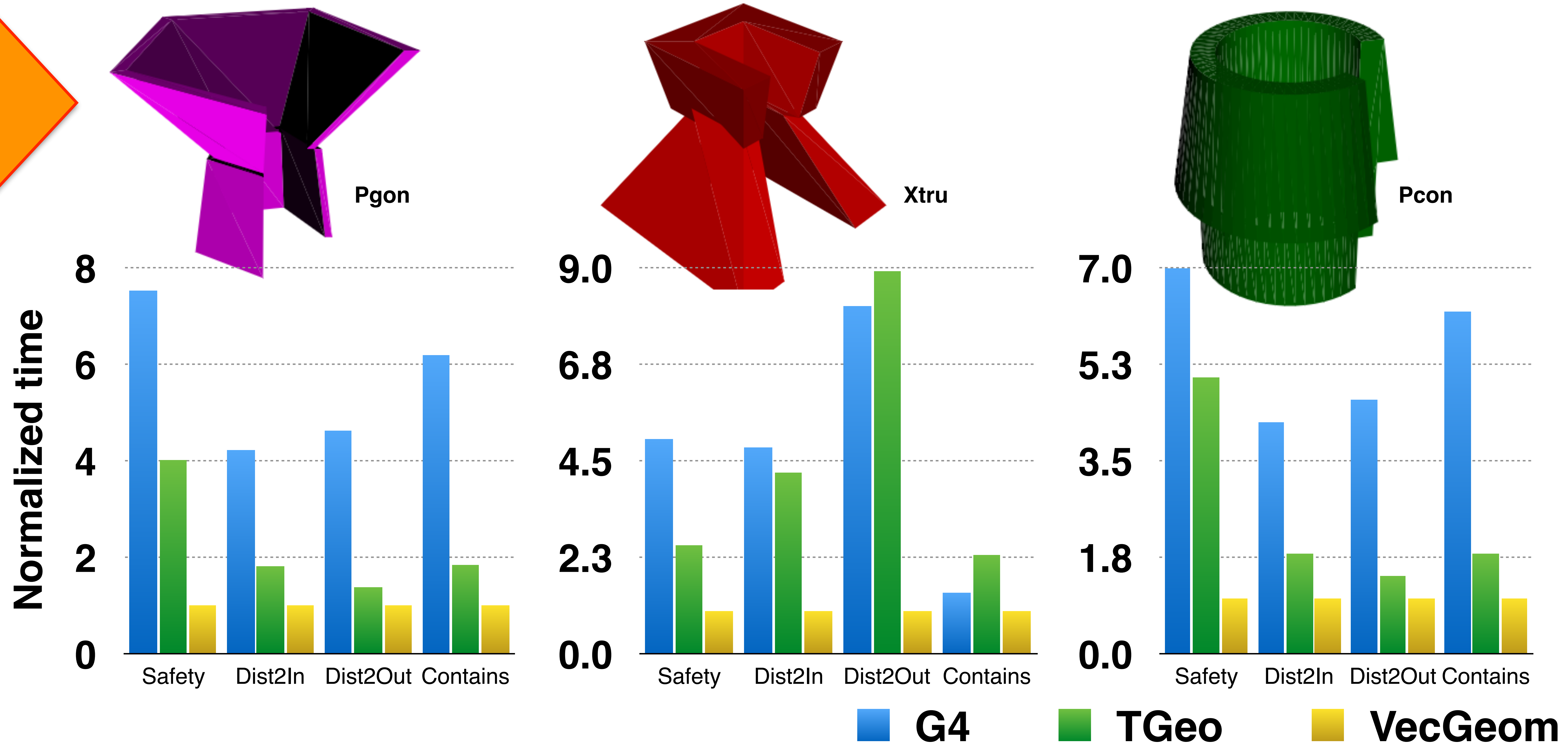
Shape-Primitives Status: The ALICE Use-Case

- VecGeom now has **all shape-primitives** to satisfy needs of most HEP experiments (Xtruded added recently)
- For ALICE simulations (Pb-Pb), demonstrate that VecGeom offers very **significant performance** gains for **the most CPU relevant shape-primitives** even in scalar track mode

up to **9x faster** than existing code

Primitive	Safety	Dist2In	Dist2Out	Contains	CPU% Sum
Pgon	2.05	2.52	0.18	1.18	**5.93**
Xtru	0.56	0.68	0.20	1.81	**3.25**
Pcon	1.07	0.32	0.05	0.13	**1.57**

% of CPU cost of shape primitives (TGeo) in typical ALICE Pb-Pb simulation



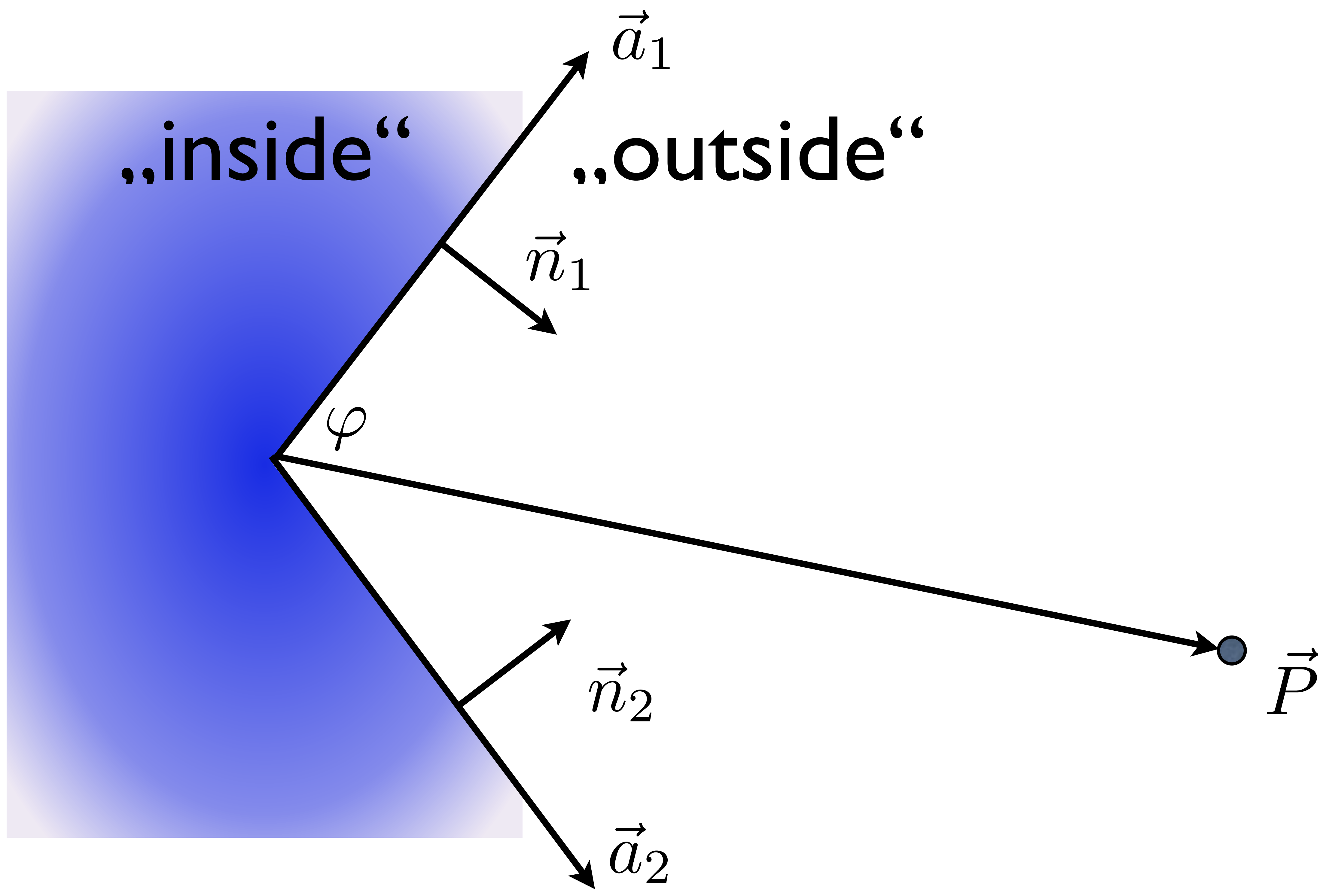
- Depending on experiment, **a few % in CPU simulation cost gainable** by switching to VecGeom primitives (integration effort into G4/TGeo under way; see separate talk)

New Features in VecGeom: Some Reasons For Improvements

- Algorithmic improvements
- Increased logical decomposition of kernels
- Increased pre-computation/caching
- Use modern C++; promote inlining; promote better compiler optimization
- Explicitly targeting inner SIMD acceleration where appropriate
- Template shape specializations
- Placement shape specializations

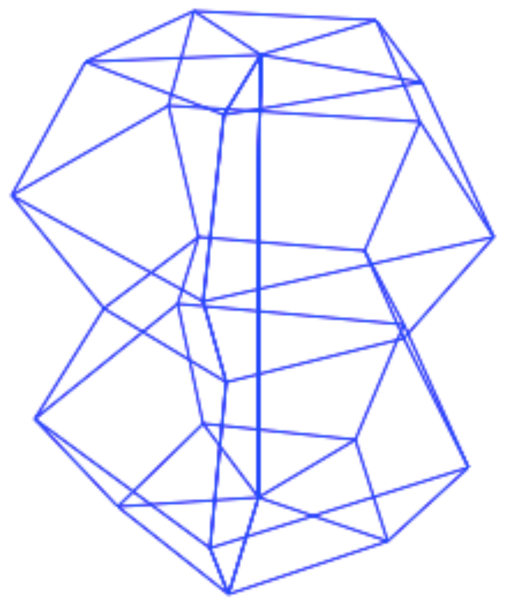
Highlight I: Algorithmic Improvement Example

- Introduced Wedge class (half-space given by phi angle)
- Logical part of many shapes: tube-segments, cone-segments, pcon-segments
- Very simple but effective improvement over existing code in USolids and G4

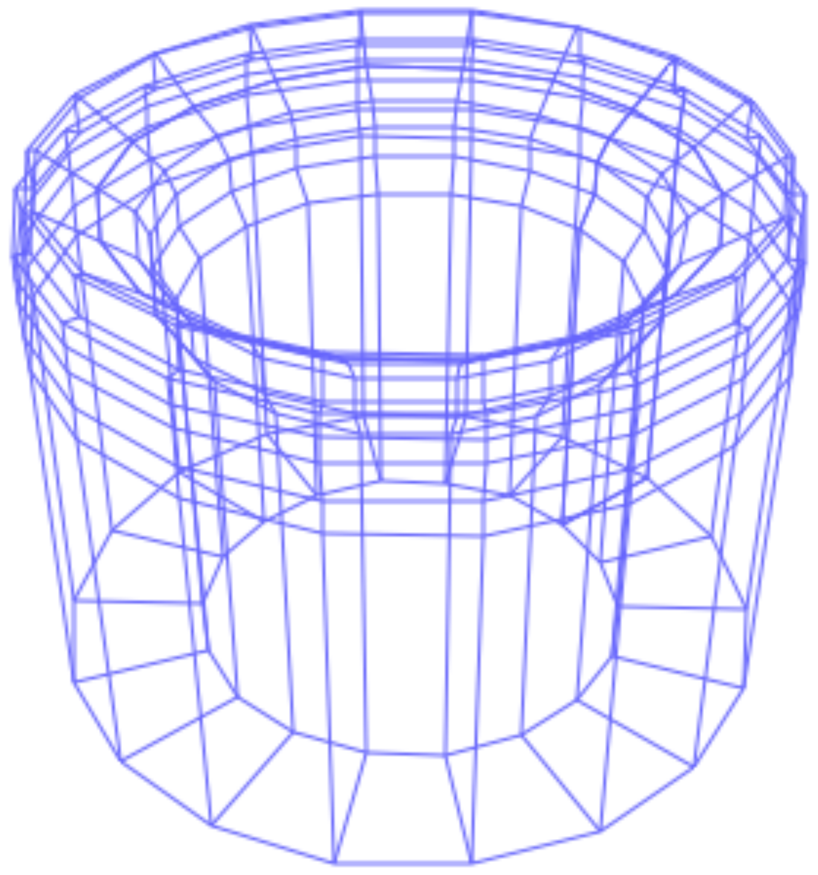


- outside test for point P was so far exclusively done using **atan2**
- now very fast test using only 2 dot products of 2D vectors
- enormously speeding up „Contains“, Safety, ... for many primitives

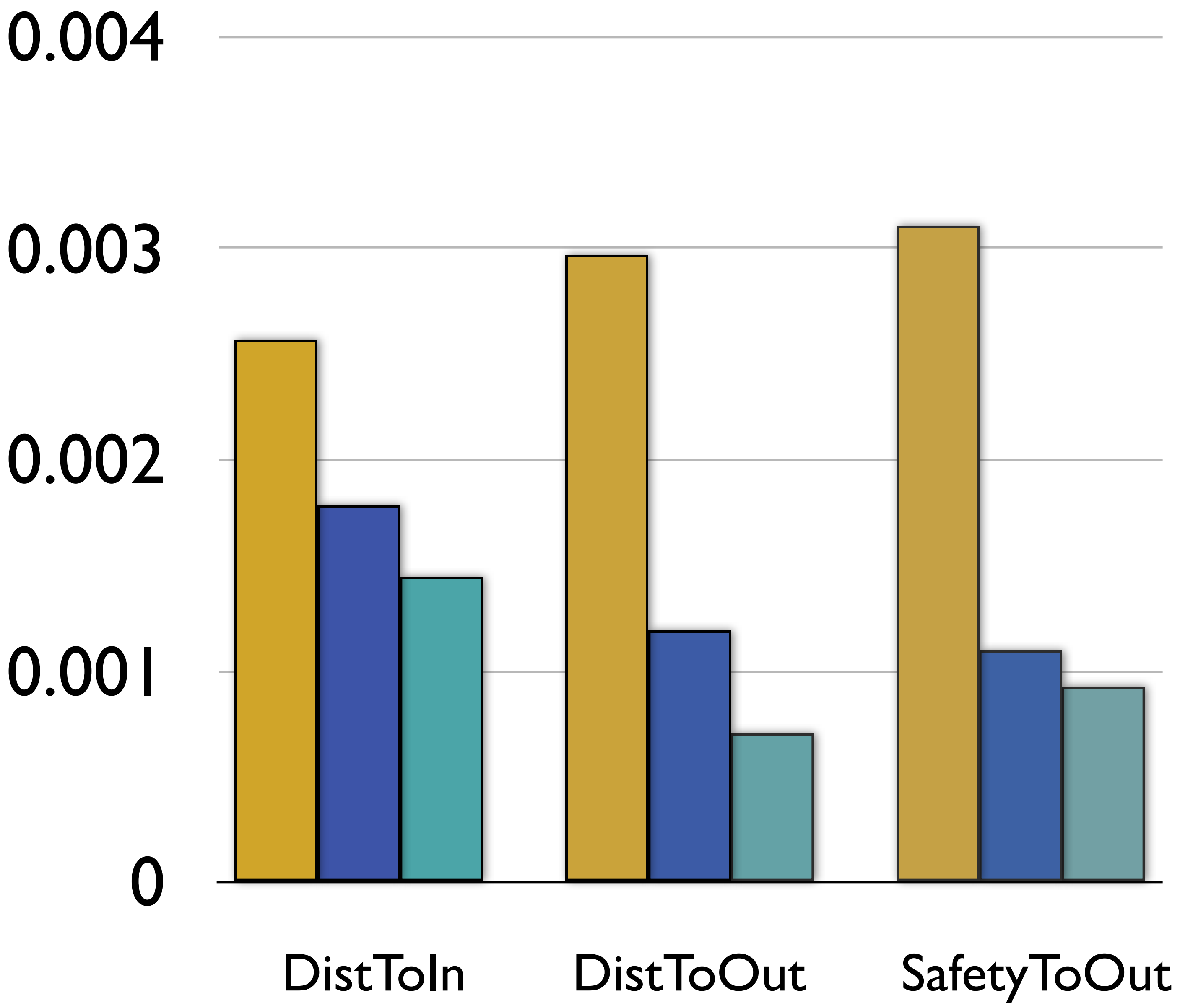
Highlight II: Internal Vectorization Example

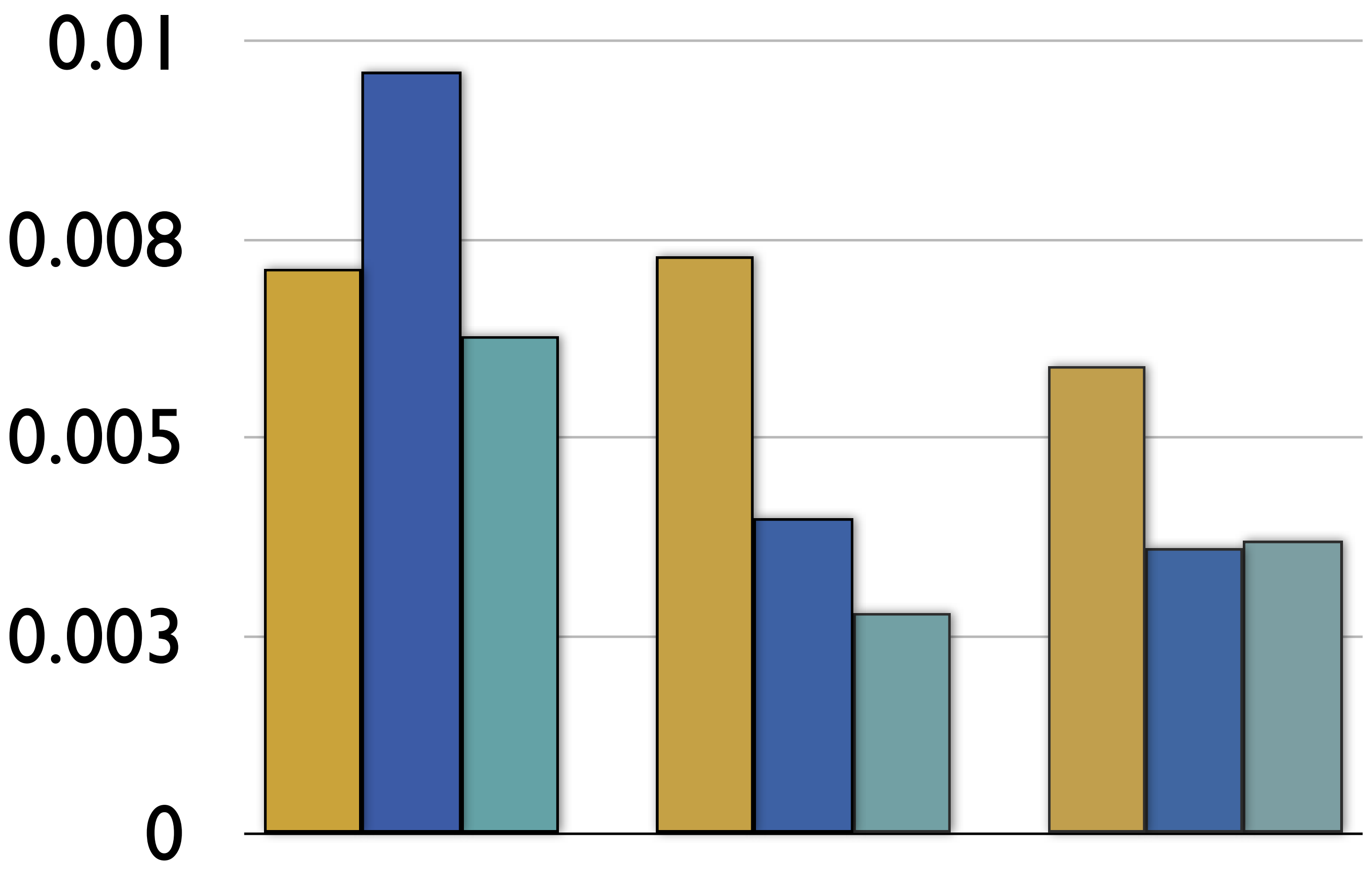


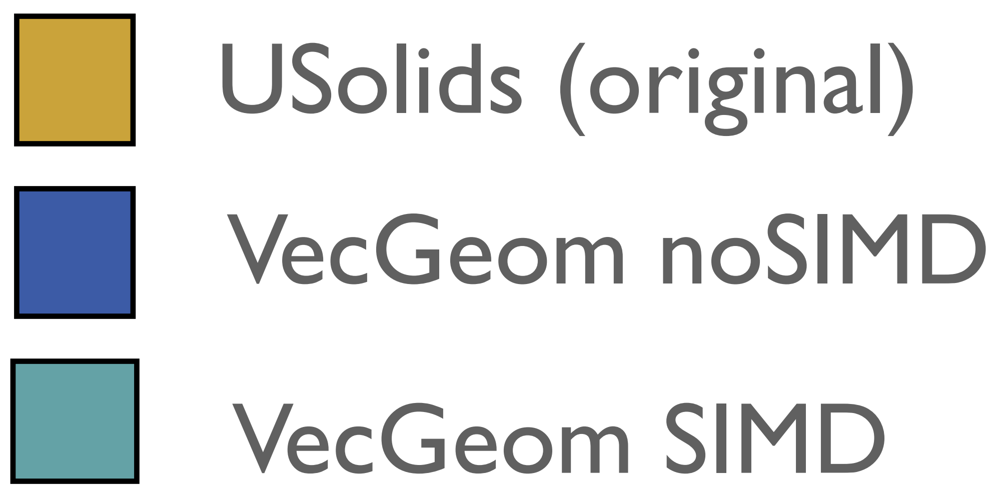
small test



HBHalf@CMS







- VecGeom implements polyhedron using internal vectorization (over facets)
- **demonstrated gain from internal vectorization** (typical factor 1.4 ish); measured on AVX
- further speedup options exist by using hierarchical SIMD accelerated trees for polyhedron (see navigation part later)

Navigation Module

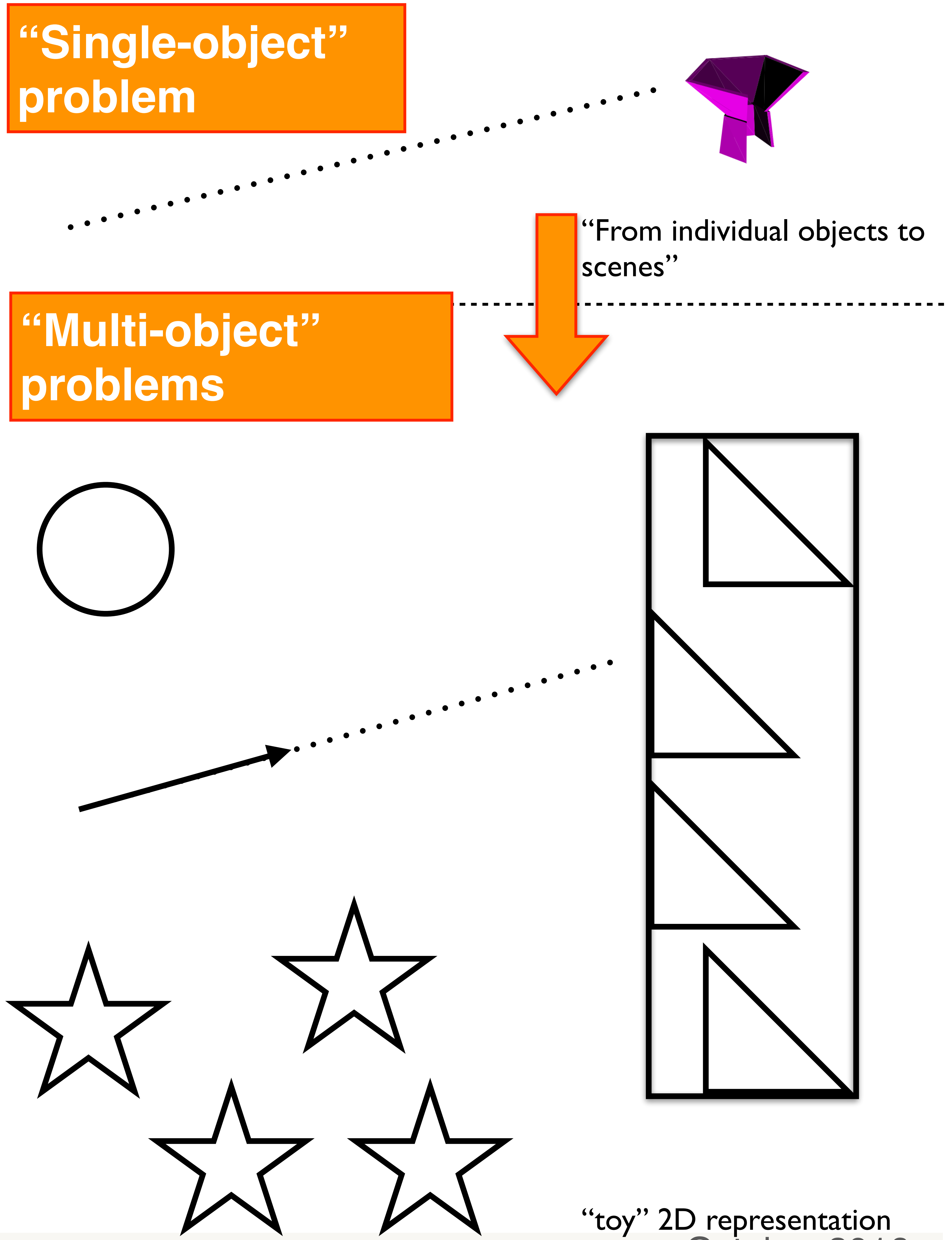
The Navigation Module

- Geometry primitives provide algorithms for simple ray - shape problems (focus on individual object)

The Navigation Module

- Geometry primitives provide algorithms for simple ray - shape problems (focus on individual object)

- **Navigation module** provides **“multi-object”** algorithms:
  - provides next colliding object + distance in a “multi-object” scene
  - provide object after the next boundary crossing
  - simulations spend significant time in navigation module (ALICE ~30% with TGeo, similar in CMS, …)

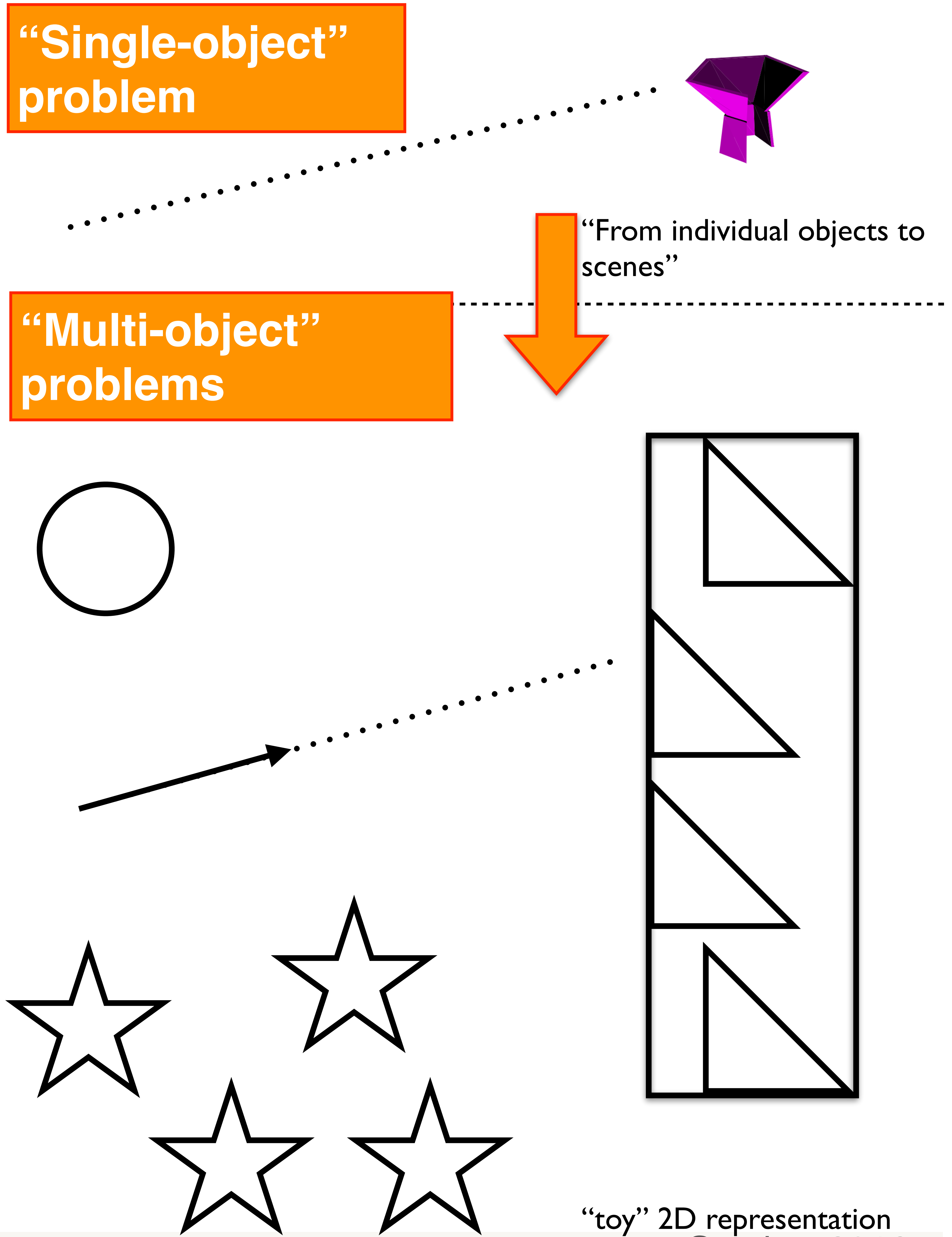


The Navigation Module

- Geometry primitives provide algorithms for simple ray - shape problems (focus on individual object)

- **Navigation module** provides “**multi-object**” algorithms:
  - provides next colliding object + distance in a “multi-object” scene
  - provide object after the next boundary crossing
  - simulations spend significant time in navigation module (ALICE ~30% with TGeo, similar in CMS, …)

- **Goals / Targets:**
  - Implement navigation system in VecGeom scaling to many particles and many threads
  - **Implement acceleration structures for fast candidate rule-out (scaling ~log(N) - see voxel techniques of G4/ TGeo)**
  - **Target explicit SIMD acceleration**



Navigation Implementation Status

- Stateless navigator classes implementing abstract interface
  - stateless for easy threading and switching tracked particles (big difference to G4; TGeo)
  - abstract interface for convenient navigator specialization for various contexts
- State (“NavigationHistory”, etc.) is always carried in separate NavigationState objects
- Templated (policy based) component-oriented implementation of navigation; easy to add new navigation algorithms or extend existing one
- Support for navigation in assemblies

SIMD Acceleration of „Voxel“ Navigation

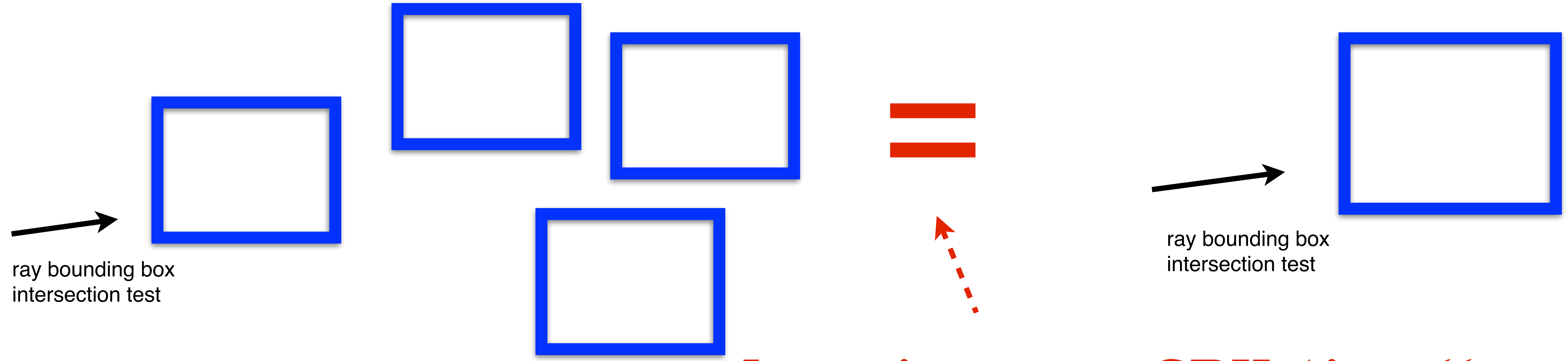
- Canonical solution for fast hit-detection: tree structures, lookup structures, bounding boxes, …
- **How to combine this with SIMD paradigm?**

SIMD Acceleration of „Voxel“ Navigation

- Canonical solution for fast hit-detection: tree structures, lookup structures, bounding boxes, …
- **How to combine this with SIMD paradigm?**
- Followed idea based on using (aligned) bounding boxes of geometry objects to filter good hit candidates

SIMD Acceleration of „Voxel“ Navigation

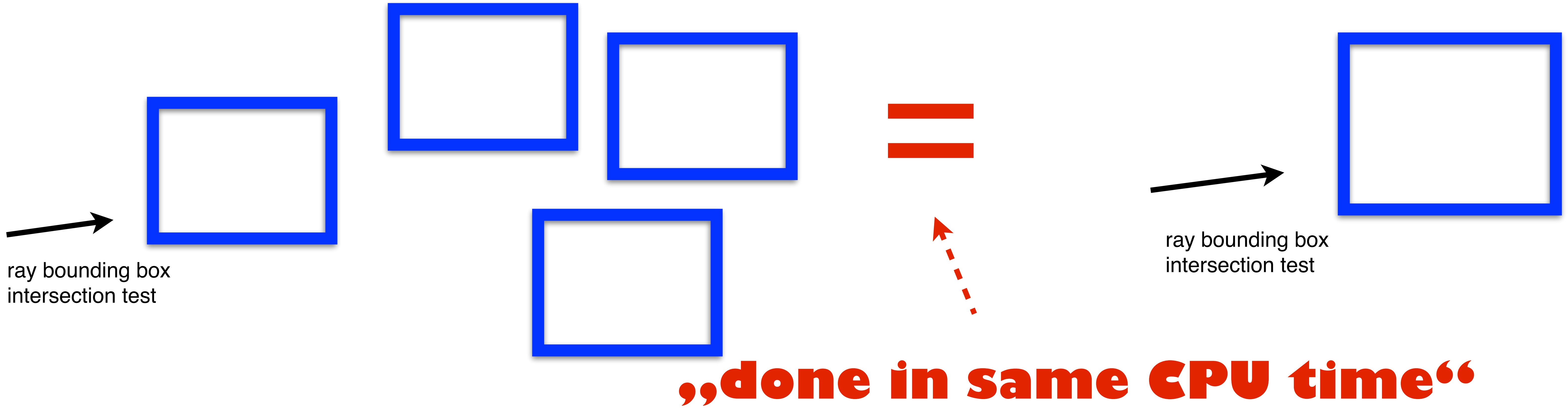
- Canonical solution for fast hit-detection: tree structures, lookup structures, bounding boxes, …
- **How to combine this with SIMD paradigm?**
- Followed idea based on using (aligned) bounding boxes of geometry objects to filter good hit candidates



„done in same CPU time“

SIMD Acceleration of „Voxel“ Navigation

- Canonical solution for fast hit-detection: tree structures, lookup structures, bounding boxes, …
- **How to combine this with SIMD paradigm?**
- Followed idea based on using (aligned) bounding boxes of geometry objects to filter good hit candidates



get **SIMD gain** from treating **group of boxes** in parallel

get **scaling** from **hierarchies** of bounding box groups (forming **regular trees**)

Inspired from e.g.: Shallow bounding volume hierarchies for fast SIMD ray tracing of incoherent rays + CPU ray-tracing libraries: Intel Embree, …

Regular Tree Building via Clusterization

- **Basic algorithm:**
  - let S == elements in SIMD register
  - cluster objects into groups of S elements (we use a variation of k-means)
  - identify bounding boxes of grouped objects as daughters of a tree node
  - iterate this process

Algorithm illustrated here for SSE (= 2 double numbers per register)

SIMD-Trees: Status + Local Benchmark

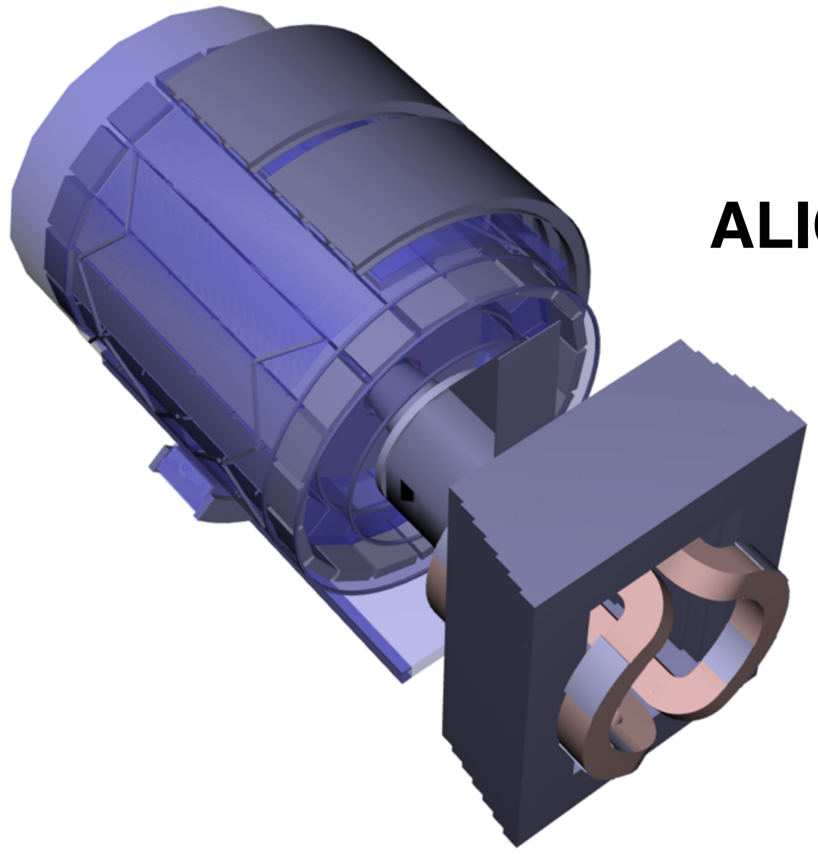
- Test approach on various detector volumes
  - most important complex volumes from ALICE: ALIC + TPC_Drift
  - a complex volume from CMS: MBWheel (~600 daughter volumes)

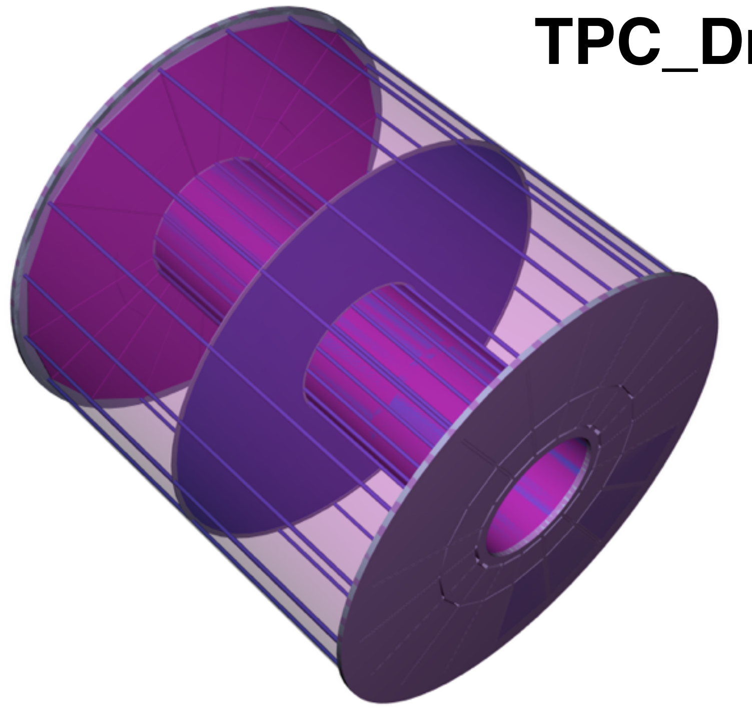
- Perform local navigation benchmark: One step + boundary crossing in the given volume for 0.5 million different tracks

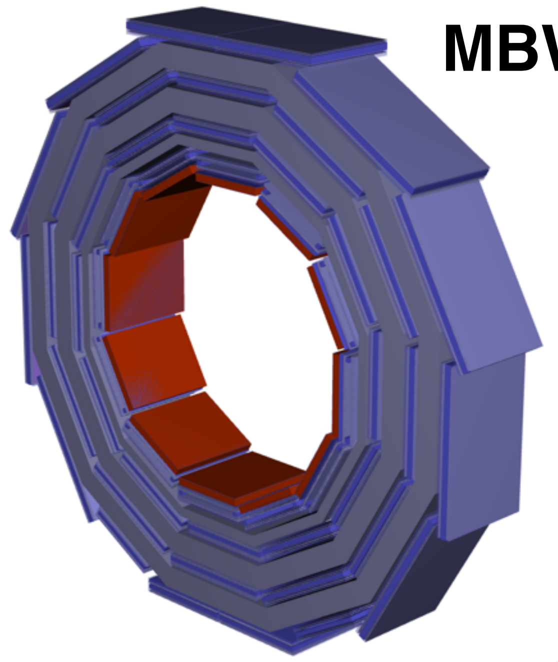
Volume	Daughters	G4	TGeo	VecGeom (SSE4.2)	VecGeom (AVX2)
ALIC (ALICE)	65	0.74	1.07	0.30	0.23
TPC_Drift (ALICE)	641	14	2.2	1.2	0.9
MBWheel (CMS)	~600	0.84	1.09	0.49	0.35

numbers are time in seconds; **worst is red**; **best is blue**

- Demonstrating **overall speedup >2** compared to existing solutions + **gain from SIMD unit** (see change SSE4.2 to AVX)



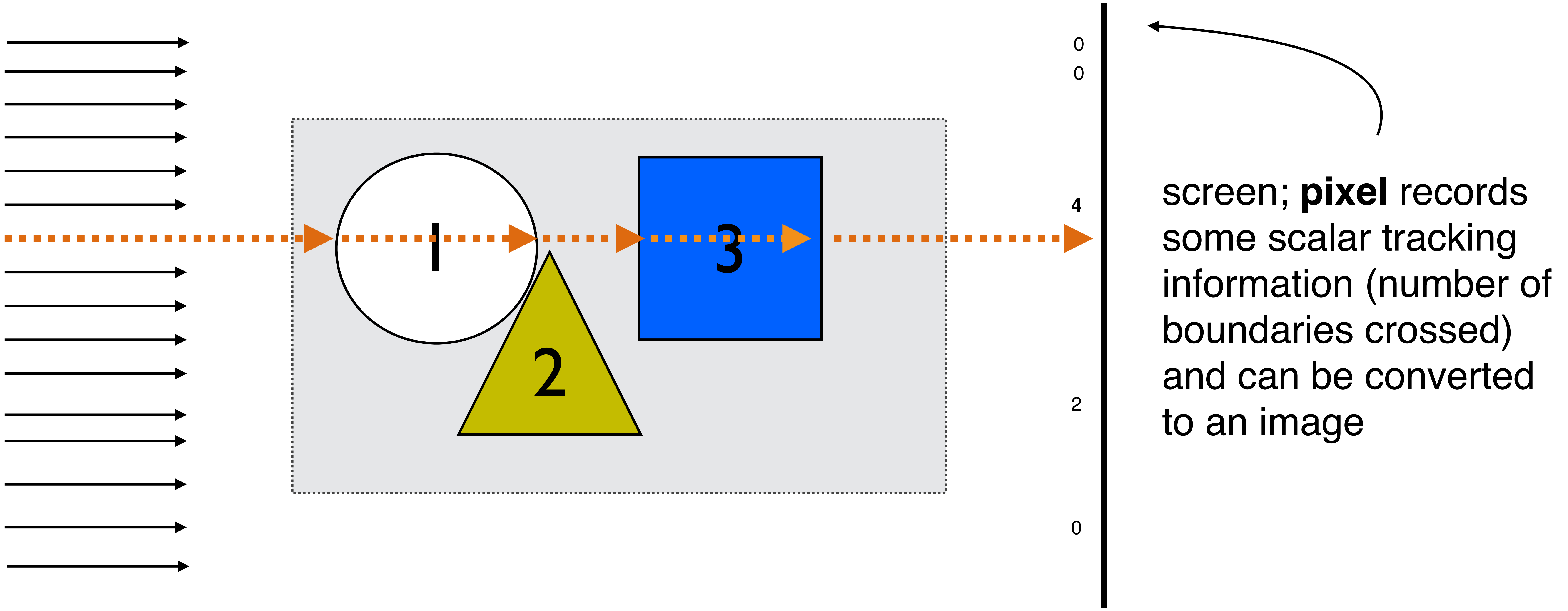




Test/Global Benchmark of VecGeom Navigation

- Evaluate VecGeom (solids + navigation) on complex modules for **multiple steps**

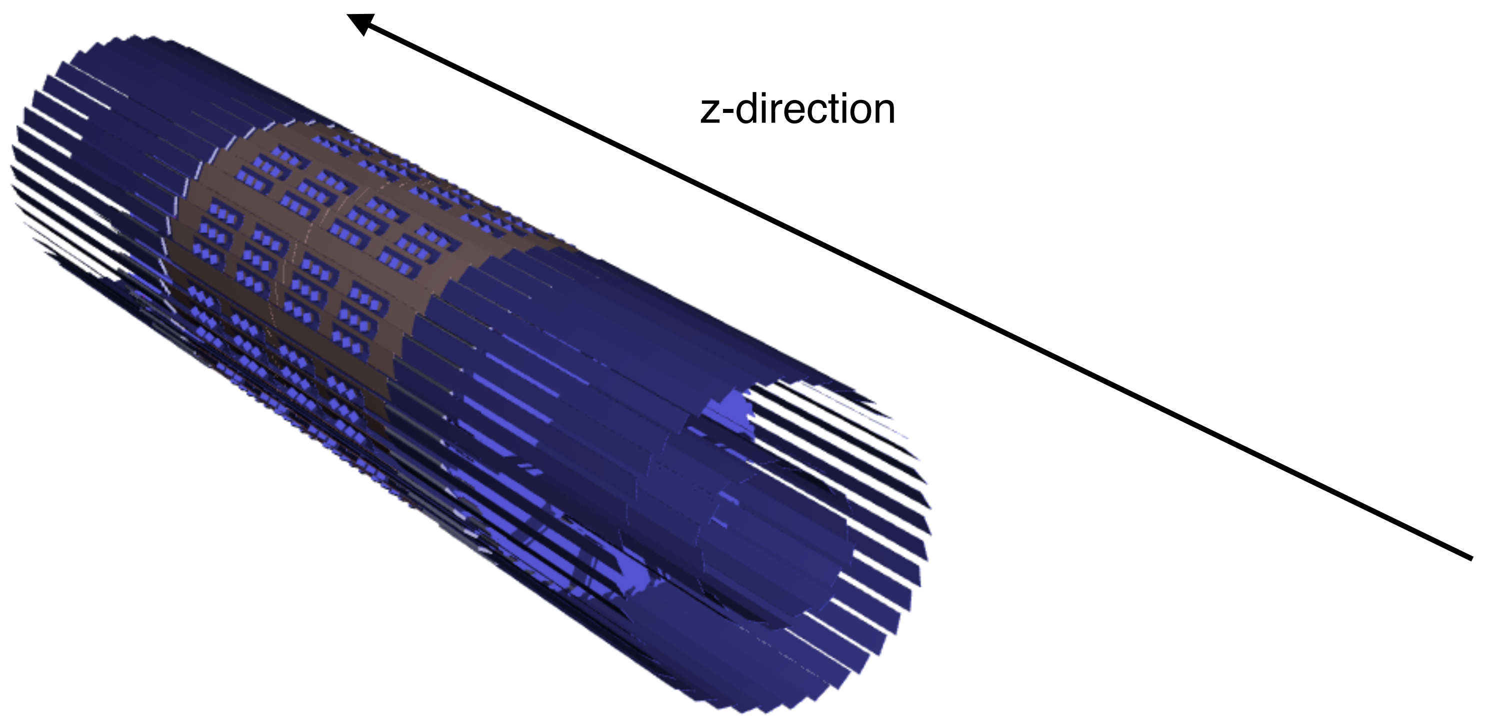
- **XRay-Benchmarker**:
  - follow geantions through geometry pixel-by-pixel
  - record some information on screen



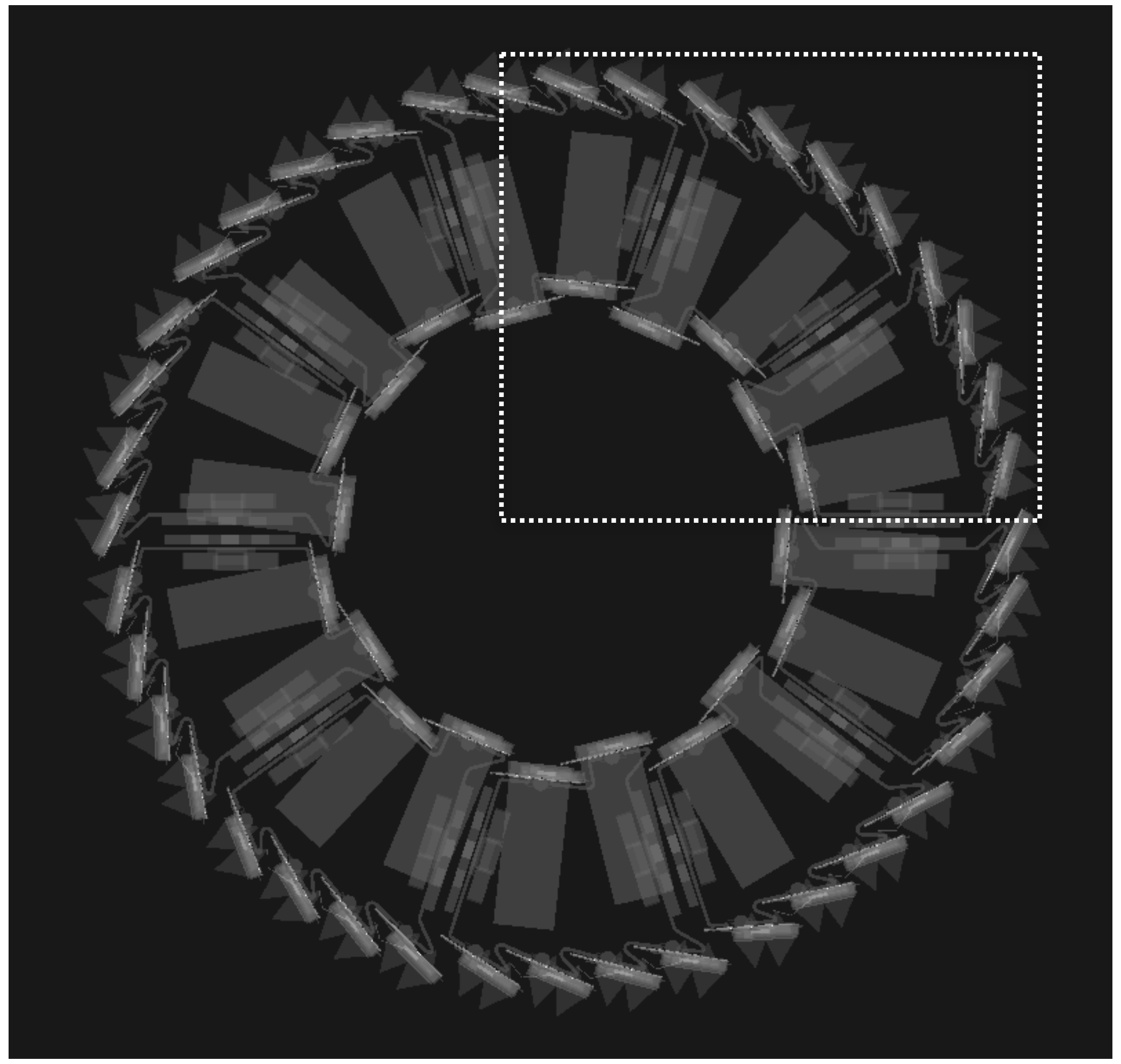
- **Perfect for validating** navigation algorithms (can do same with G4/TGeo)

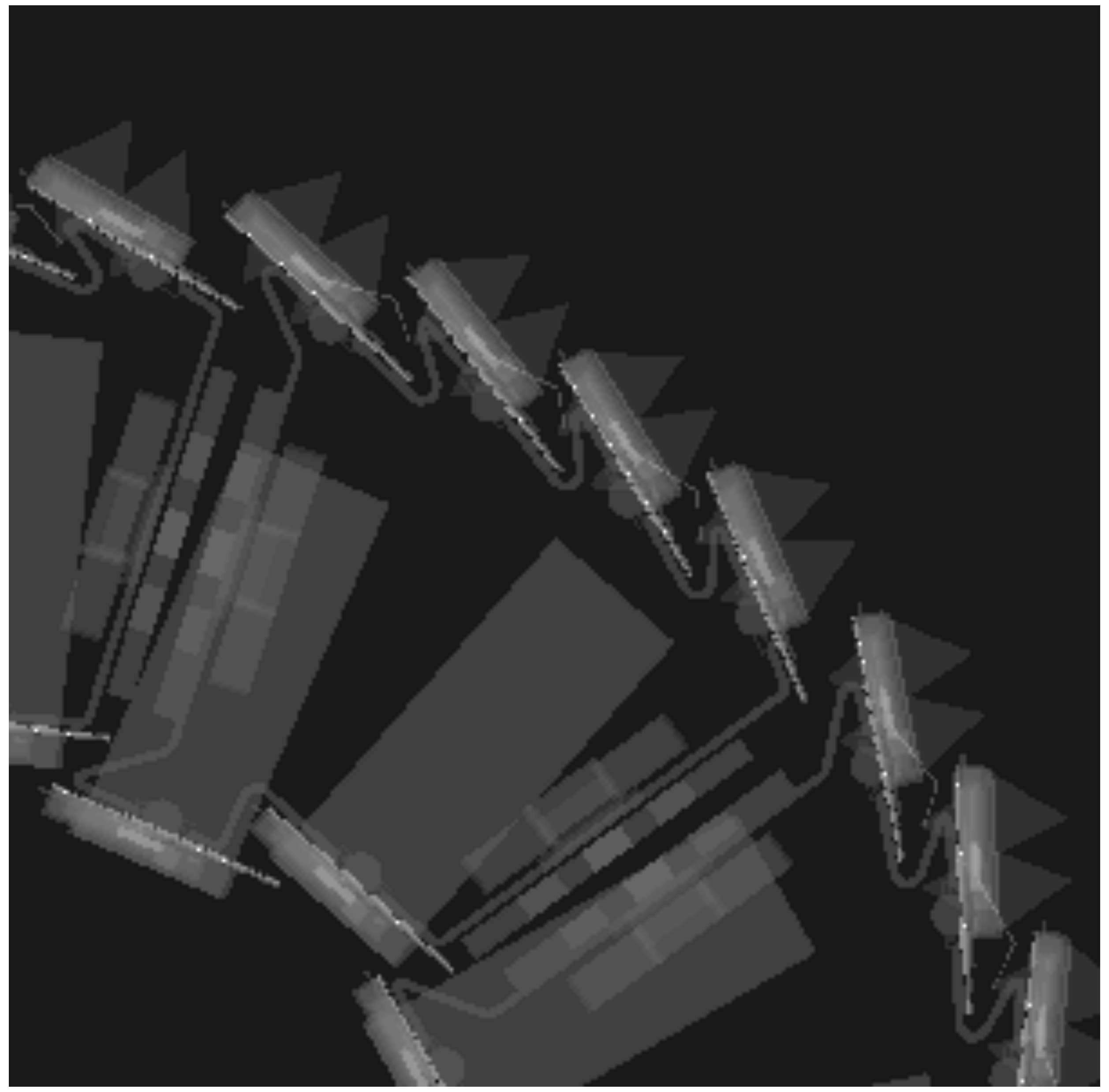
- **Good to get a global idea of library performance**

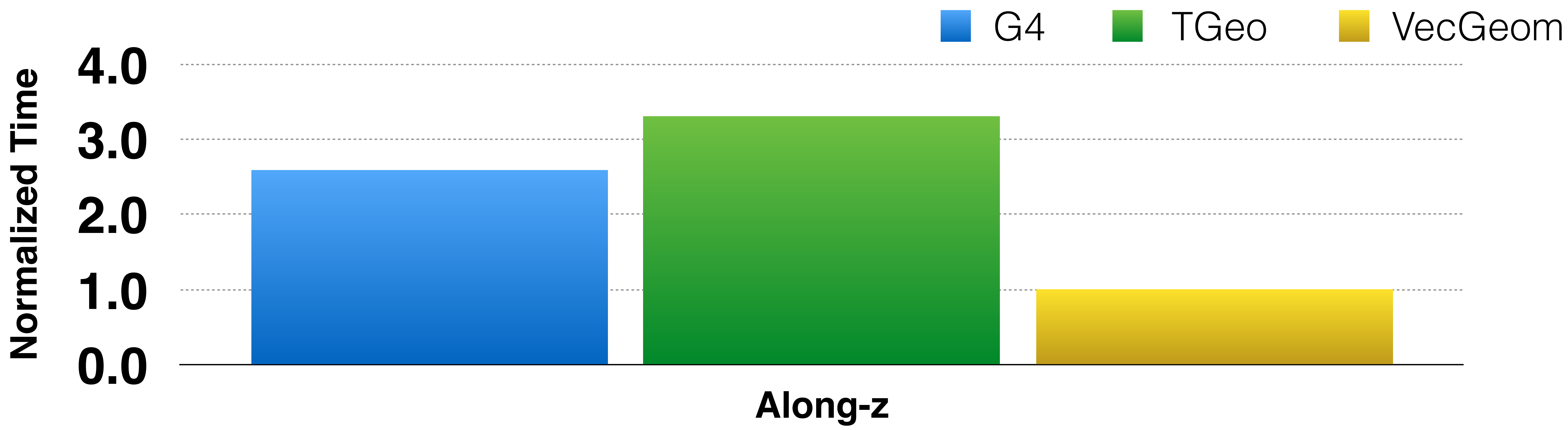
Geantino-XRay: Global Performance



- Example for ALICE ITSSPD module (to test assembly implementation)
- Perfect agreement between G4/ TGeo/VecGeom
- Observe generally **factors > 2.6x** speed improvement against other packages
- Another indication of global performance advantage of VecGeom

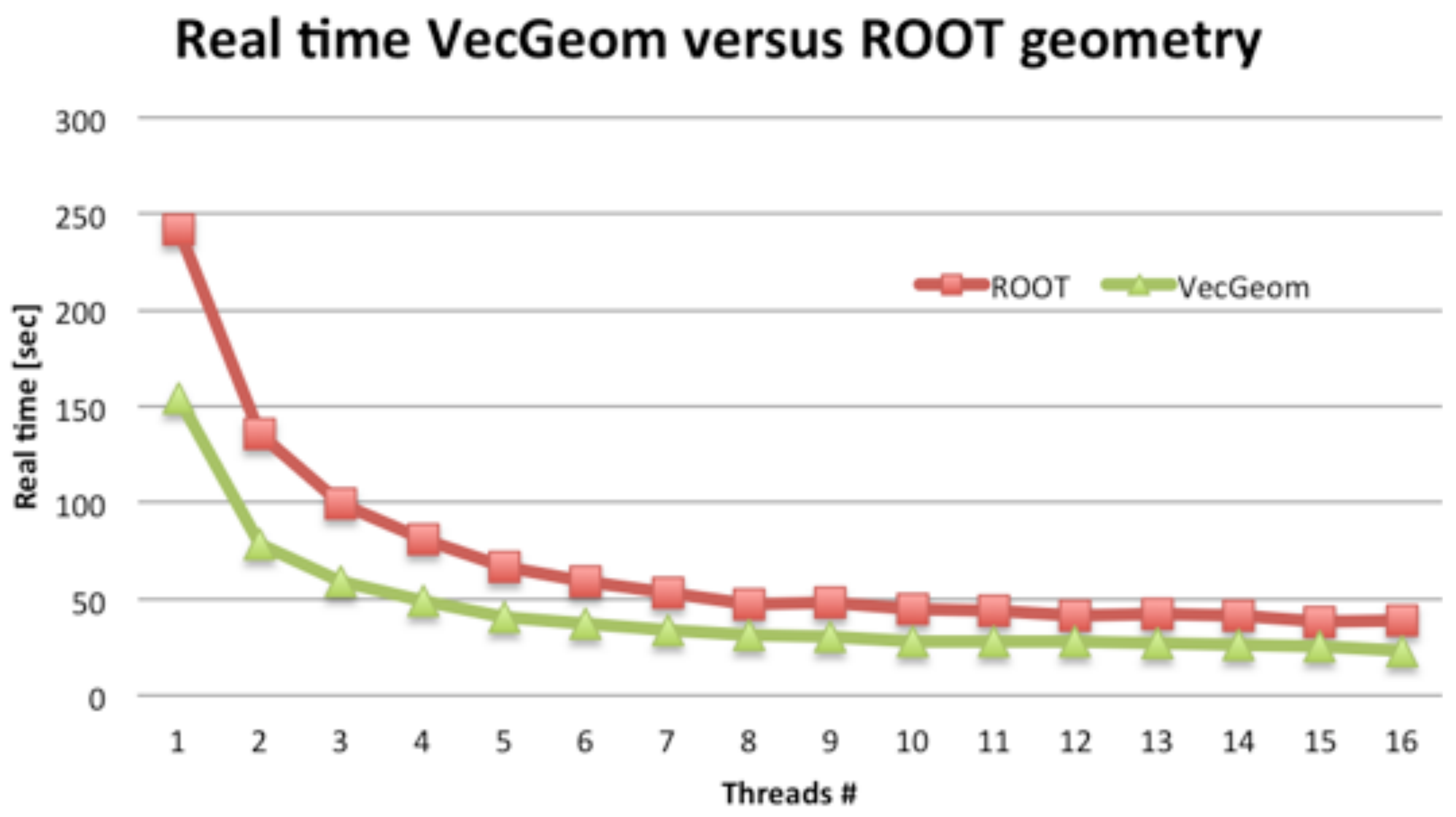


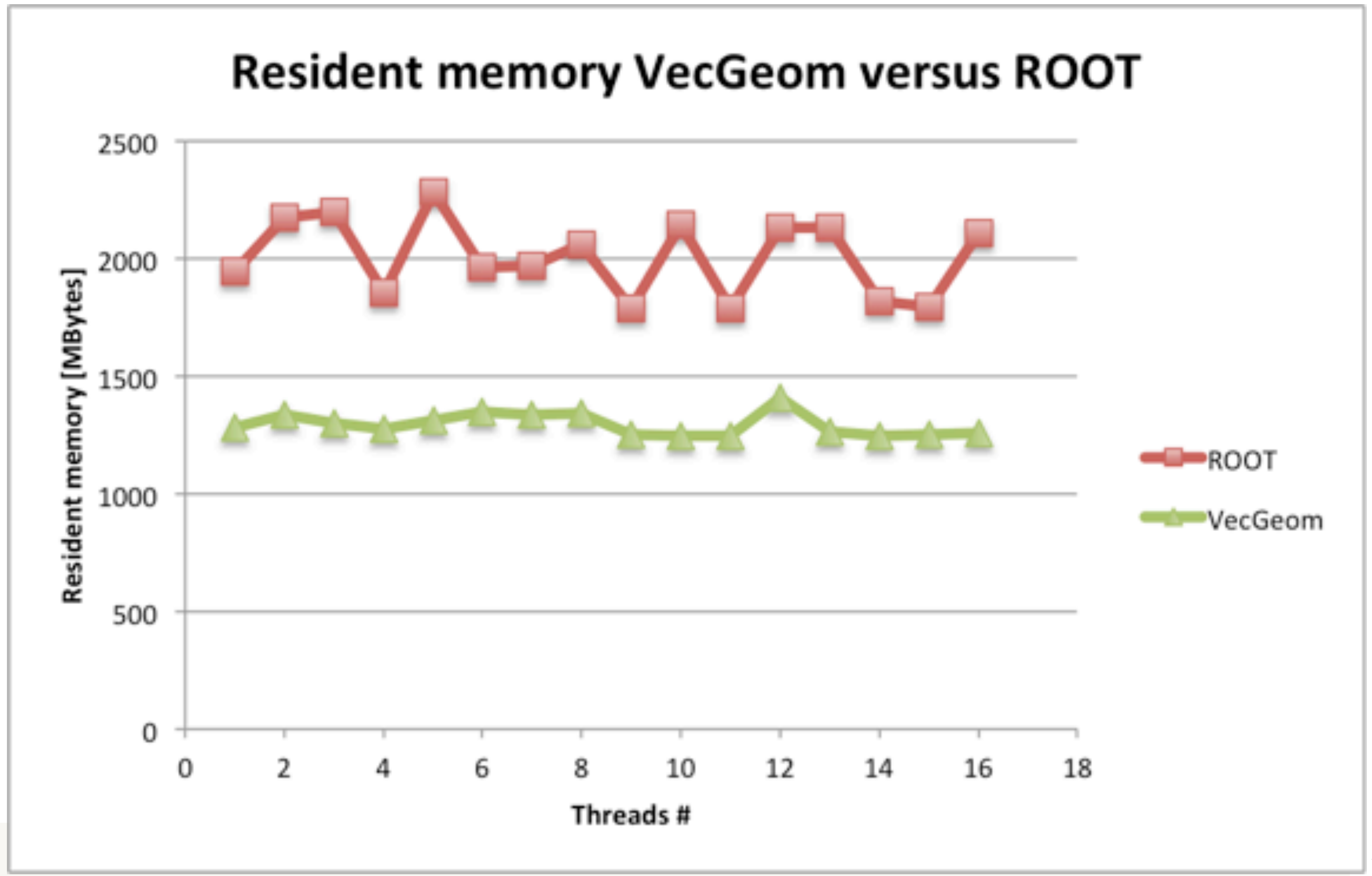




Geant-V: Switching TGeo to VecGeom

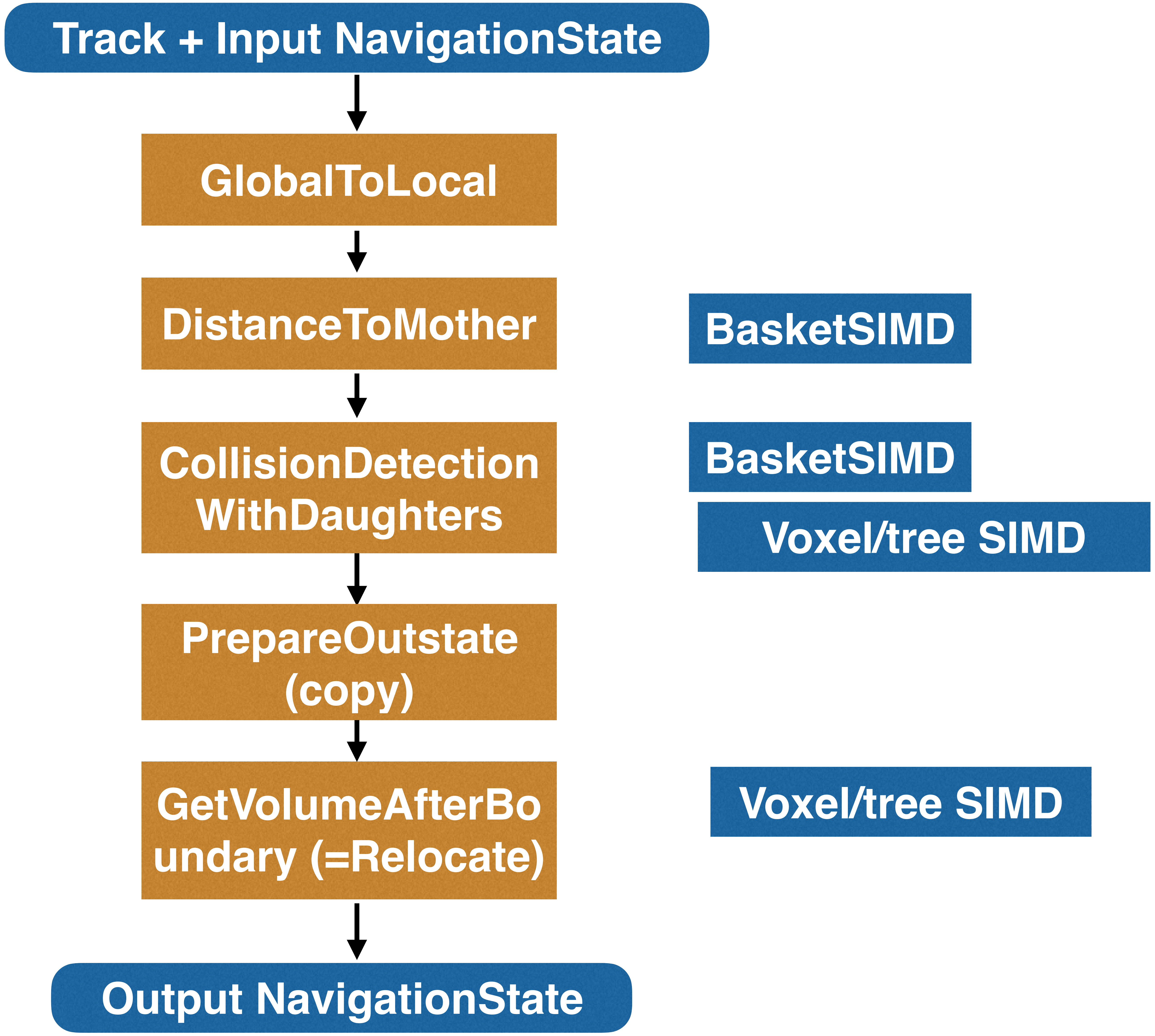
- Can globally benchmark VecGeom within full Geant-V simulation (with tabulated physics) by switching between TGeo <-> VecGeom
- Gain a **factor ~1.6 in simulation runtime** for CMS benchmark using VecGeom
- **Consuming considerably less memory**
- Gain from basket treatment still under investigation (see next slide)





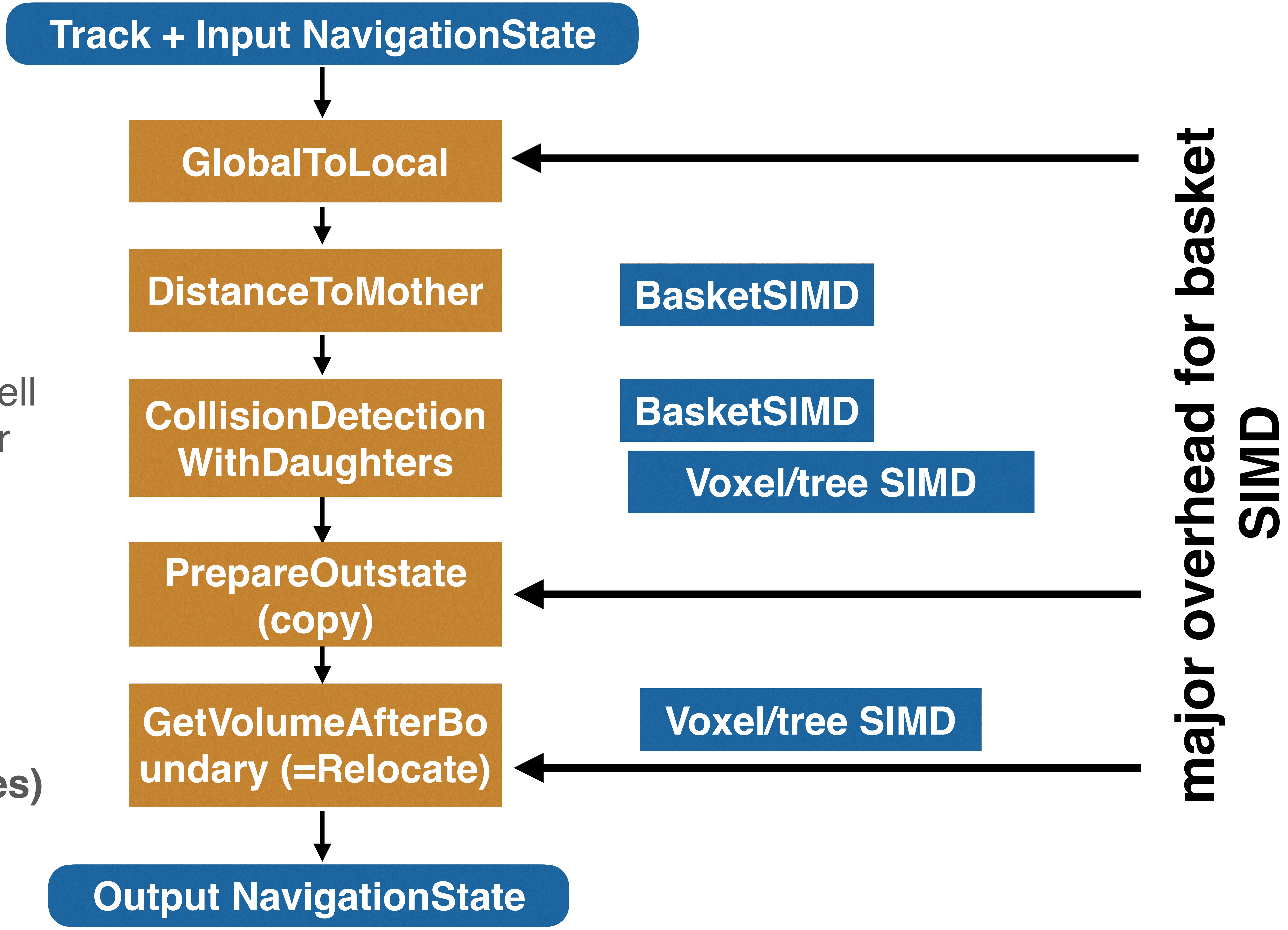
Remaining Challenges

- Achieved SIMD acceleration in various parts of full navigation



Remaining Challenges

- Achieved SIMD acceleration in various parts of full navigation
- Full SIMD gain in basket mode remains a challenge because some algorithmic parts do not vectorize well in basket mode and represent major overheads
- **We need to reduce these overheads !**
- **Currently addressing these challenges in R&D (see next slides)**



Further R&D in Navigation Optimization

- Traditional navigation algorithms (in G4/TGeo world) are still

Further R&D in Navigation Optimization

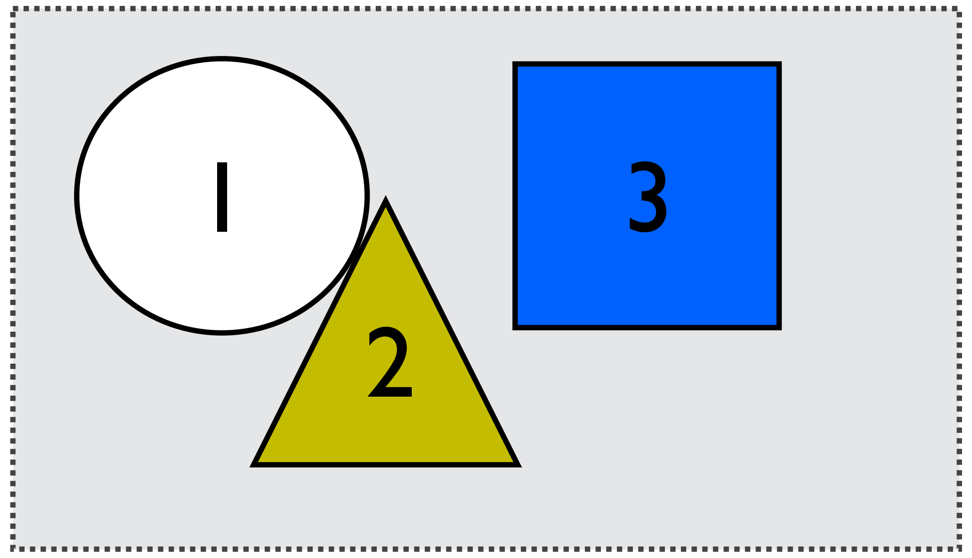
- Traditional navigation algorithms (in G4/TGeo world) are still
- … to generic
  - **runtime polymorphic** approach
  - e.g., no internal vectorization over primitives possible

Further R&D in Navigation Optimization

- Traditional navigation algorithms (in G4/TGeo world) are still

- … to generic
  - **runtime polymorphic** approach
  - e.g., no internal vectorization over primitives possible

- … poorly exploiting structural and static information about a scene
  - **no usage of boundary touching** relation between objects
  - **no fast lookup of global-local transformation** for placed entities
  - etc…

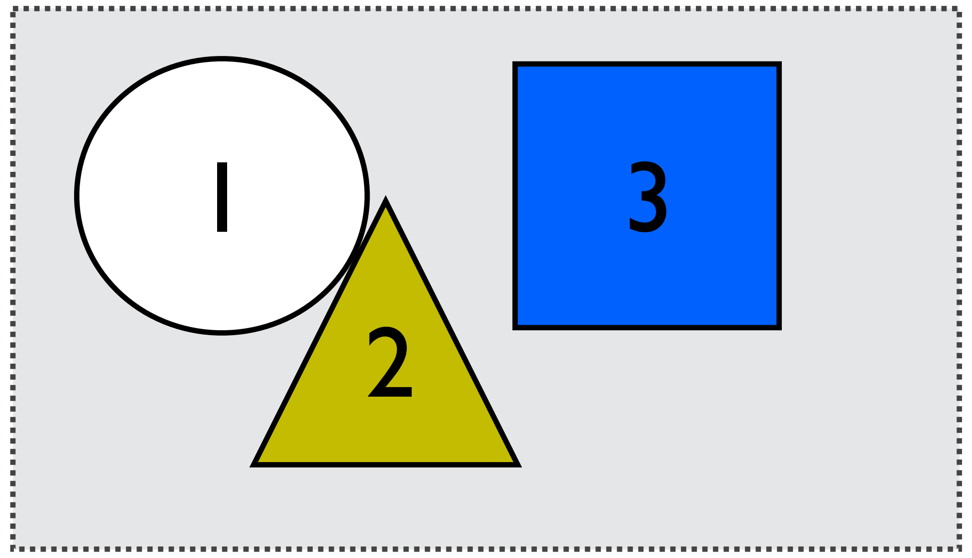


$$\begin{array}{c c} & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}$$

Example: Static geometry analysis can reveal that object1 only touches object2; tracks leaving 1 never have to be checked against 3 for relocating

Further R&D in Navigation Optimization

- Traditional navigation algorithms (in G4/TGeo world) are still
- … to generic
  - **runtime polymorphic** approach
  - e.g., no internal vectorization over primitives possible
- … poorly exploiting structural and static information about a scene
  - **no usage of boundary touching** relation between objects
  - **no fast lookup of global-local transformation** for placed entities
  - etc…



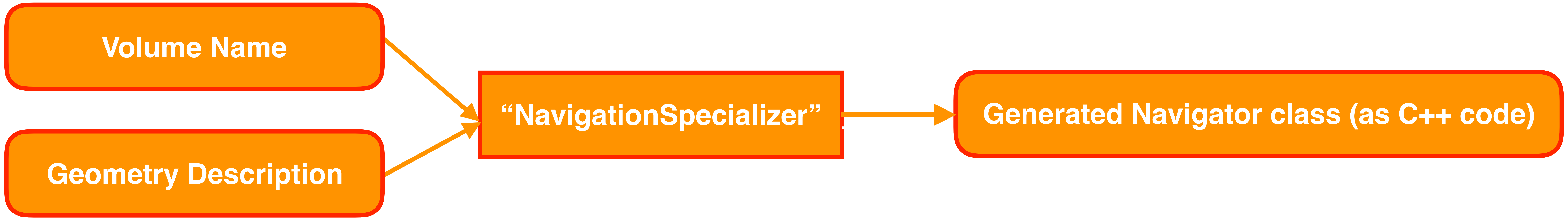
$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}$$

Example: Static geometry analysis can reveal that object1 only touches object2; tracks leaving 1 never have to be checked against 3 for relocating

- HEP detectors are pretty static objects; most things are known at compile time or constant during (long) run-time
- Opportunity to pre-analyse + pre-compute + compile-time optimize
- **Goal:** Exploit these opportunities via **volume-specialized navigator algorithms** produced via automatic C++ code generation

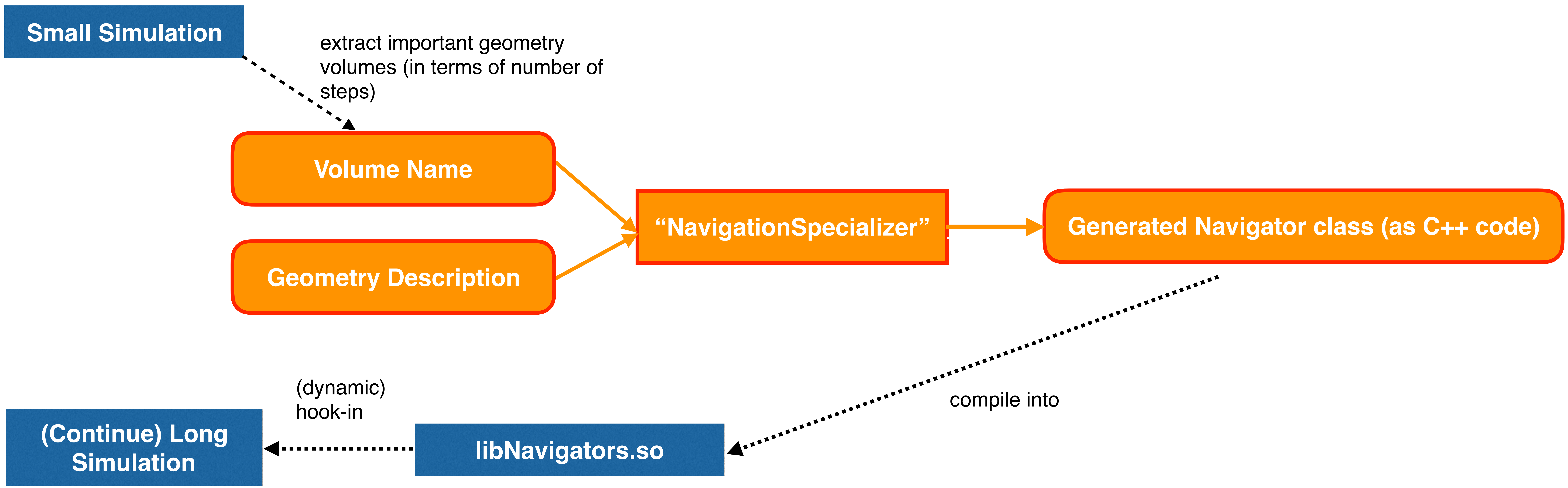
Implementation Status and Workflow

- Prototype service to generate volume-specialized navigator algorithms has been implemented
  - considerably reduced virtual functions
  - reduce time spent in coordinate transformation (via compile-time lookup structures)
  - put static neighbourhood information for fast relocation
  - reduce time in copying state information for navigation …



Implementation Status and Workflow

- Prototype service to generate volume-specialized navigator algorithms has been implemented
  - considerably reduced virtual functions
  - reduce time spent in coordinate transformation (via compile-time lookup structures)
  - put static neighbourhood information for fast relocation
  - reduce time in copying state information for navigation …
- Can be embedded into a (JIT) workflow of a simulation



Benchmarking Specialized Navigators

- Extracted important ("showering") volumes (in terms of number of steps) in an ALICE Pb-Pb simulation and measured time to do one "step" in these volumes

preliminary

Volume	G4	TGeo	VecGeom Normal	VecGeom Specialized	EXTRA SPEEDUP
ZNST	0.24	0.28	0.10	0.06	1.67
ZPST	0.25	0.29	0.11	0.06	1.83
DCML	0.24	0.28	0.12	0.06	2.00
voRBCuTube	0.16	0.24	0.10	0.06	1.67
ZNGx	0.09	0.18	0.06	0.03	2.00
AFaGraphiteCone	0.74	0.36	0.11	0.03	3.67

numbers are time in seconds; **worst is red**; **best is blue**

- **Navigator specialization** delivers **extra speedup kick**; making gain compared to G4/TGeo even more significant

Effect of Specialization on Basket SIMD

- Navigator specialization **boosts** gain from **SIMD basket interfaces** (in simple setups)
- Seen from better ratio scalar/vector for specialized timings

preliminary

volume	base algo	normal scalar	normal vecor	specialized scalar	specialized vector
HVQX	simple	**12.6**	**10.6**	**6.4**	**4.7**
ZDC_EMFiber	simple	**10.1**	**8.8**	**5.9**	**2.6**
ZDC_EMLayer	voxel (hybrid)	**27.0**	**27.0**	**19.7**	**19.3**

numbers are time (in some units) doing a navigation step; volumes are important showering volumes identified in a Geant-V simulation of CMS

Conclusion

- VecGeom is a full multi-platform and multi-API geometry system with the potential to serve both Geant-V as well as Geant-4
- VecGeom makes use of SIMD opportunities in various contexts and shows considerable performance (CPU + memory) benefits compared to existing solutions
- Demonstrated avenue for further performance opportunities (to be put in production)

Future plans

- R&D work to further accelerate various algorithmic parts
  - Continue navigator specialization work
  - Acceleration of tessellated solid + multi-union using the ideas used in navigation module
  - Learn more from developments in ray-tracing libraries (Intel Embree, …)
- Implement missing geometry primitives
  - Twisted primitives, …
- Consolidation of code / API / tests
- Native connection to Geant4

Backup

Testing VecGeom

- Unit tests
- Consistency test
  - ShapeTester
- Verification against existing packages (benchmarker) + performance tests
  - XRayBenchmarker
  - Benchmarker
- Test through regression / unit tests of Geant4 (see talk on integration)

Some details on benchmark environment

- Benchmark setup:
  - All benchmarks presented here were run with tag “W40-16” of VecGeom
  - Benchmark machine: Intel(R)-Core(TM) i7-5930K running CERN CC7
  - Compiler gcc4.8.5
  - Vc 1.2.0 backend of VecCore with native (=AVX2) instruction set (unless otherwise specified)

Geometry-Primitive Status: The Big Matrix

- Performance status for all geometry primitives used in ALICE
- G4S = speedup of VecGeom scalar against G4; RS = (against ROOT); US (against USolids original); SIMD = gain from basked interface
- few slowdowns mainly due to stricter conventions than before

Speedup

>2	very good	
>1.2	good	
>1.0	neutral	
<1	bit slower	

NEEDS UPDATE

	DistanceToIn				DistanceToOut				Contains/Inside			
	G4S	RS	US	SIMD	G4S	RS	US	SIMD	G4S	RS	US	SIMD
Box	0.82	1.08	0.89	2.49	1.40	1.22	1.00	2.22	0.83	1.01	1.03	1.97
Tubes	1.21	1.57	1.23	1.96	1.17	0.88	1.44	2.16	0.85	0.91	1.03	1.95
TubeSegs	1.16	1.34	1.12	2.09	1.75	1.00	1.81	2.29	2.47	1.85	1.84	2.22
TubesCombined	1.19	1.51	1.20	1.99	1.32	0.91	1.53	2.19	1.11	1.31	1.24	2.16
Cones	1.24	2.03	1.14	1.19	1.55	1.27	1.34	1.27	1.25	1.27	1.34	1.60
Booleans	4.29	1.79	-	1.04	3.63	1.90	-	1.03	4.44	1.26	-	1.15
Pcon	4.21	1.79	1.21	1.07	4.65	1.37	1.31	1.11	6.18	1.83	1.36	1.29
Pgon	2.02	5.07	2.73	1.02	1.88	4.33	2.78	1.05	7.15	5.94	5.87	1.27
Arb8	7.24	1.66	2.12	1.17	3.63	2.80	2.75	2.38	16.97	1.34	1.17	2.54
Gtra	4.02	1.13	1.35	1.31	1.07	1.35	1.38	2.04	16.52	1.17	1.08	2.56
Para	1.17	1.15	-	2.05	1.13	1.01	-	1.88	1.04	1.06	-	1.66
Trd1	1.18	1.18	1.23	3.02	1.87	1.87	1.92	2.14	0.95	1.07	0.97	4.15
Xtru	4.79	4.20	4.08	1.03	8.16	8.91	7.71	1.07	1.44	2.36	1.38	1.22
Torus	11.50	3.50	-	1.05	10.94	4.58	-	1.10	2.35	4.99	-	1.85
Trap	1.14	1.19	0.96	2.28	1.16	1.03	1.11	2.36	1.46	1.68	0.95	1.97