Running prototype of USolids using SIMD instructions

Cosmo, G. (CERN)

30 January 2017

The AIDA-2020 Advanced European Infrastructures for Detectors at Accelerators project has received funding from the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement no. 654168.

This work is part of AIDA-2020 Work Package 3: Advanced software.

The electronic version of this AIDA-2020 Publication is available via the AIDA-2020 web site <http://aida2020.web.cern.ch> or on the CERN Document Server at the following URL: <http://cds.cern.ch/search?p=AIDA-2020-MS39>

Copyright © CERN for the benefit of the AIDA-2020 Consortium
Abstract:
The USolids package [1] has been revised and extended to support vector signatures, providing the ability to perform queries on the geometrical primitives in parallel and allow for concurrent queries within the VecGeom package [2]. A running prototype of USolids within VecGeom is now available, implementing most of the shapes defining the standard set in the GDML schema [3]; SIMD (Single Instruction, Multiple Data) instructions are being used where possible, while the API has been extended to provide vector signatures. In this document, we briefly report on the design principles behind the prototype implementation.
AIDA-2020 Consortium, 2017
For more information on AIDA-2020, its partners and contributors please see www.cern.ch/AIDA2020

The Advanced European Infrastructures for Detectors at Accelerators (AIDA-2020) project has received funding from the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement no. 654168. AIDA-2020 began in May 2015 and will run for 4 years.

### Delivery Slip

<table>
<thead>
<tr>
<th>Name</th>
<th>Partner</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authored by</td>
<td>G. Cosmo [Task coordinator]</td>
<td>CERN</td>
</tr>
<tr>
<td>Edited by</td>
<td>G. Cosmo [Task coordinator]</td>
<td>CERN</td>
</tr>
<tr>
<td>Reviewed by</td>
<td>W. Pokorski [WP coordinator]</td>
<td>CERN</td>
</tr>
<tr>
<td></td>
<td>F. Sefkow [Scientific Coordinator]</td>
<td>DESY</td>
</tr>
<tr>
<td>Approved by</td>
<td>Scientific Coordinator</td>
<td></td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS

1. INTRODUCTION.........................................................................................................................4
2. DESIGN AND IMPLEMENTATION........................................................................................................5
3. CONCLUSIONS ................................................................................................................................6
4. REFERENCES ..................................................................................................................................7

ANNEX: GLOSSARY ...............................................................................................................................7
Executive summary

Most geometry primitives implemented in the USolids package [1] have been reengineered within the VecGeom package [2] to support vector signatures and make best possible use of SIMD instructions in concurrent queries. The prototype is now available with VecGeom v00.03.00 and can be used for detector simulation in Geant4 [4] and from ROOT [5].

1. INTRODUCTION

Geometry modelling is an important component in detector design and simulation; geometrical models describing the experimental setups in either simulation or reconstruction programs are using geometrical primitives as building blocks to describe detectors in the detail. The original aim of the USolids (Unified Solids) library was to provide a new software library of geometrical primitives, unifying and improving on the C++ implementations available in the Geant4 simulation toolkit and in ROOT.

Based on the USolids package, we have refactored the core geometry algorithms for the solid primitives using generic templated algorithms, and make these available in a new package, VecGeom. The current prototype (VecGeom v00.03.00) includes most shapes which define the GDML schema and extends the original set of primitives available in the USolids library. For most shapes, we could generate well performing SIMD kernels for the treatment of concurrent queries; the same generic code is used for the scalar API with improved or equivalent performance compared to the original USolids code; the same primitives code is also available as CUDA for use on GPUs.

1 CUDA is a parallel computing platform and application programming interface (API) model created by Nvidia.
2. DESIGN AND IMPLEMENTATION

Achieving reliable SIMD vectorisation in the implementation of algorithms has often required in the past adoption of platform specific code using assembly language or compiler intrinsics. In recent years, this challenge has become a more manageable task thanks to the availability of new high-level C++ (templated) wrapper libraries which hide away the complexity of SIMD hardware-instructions by providing user-friendly C++ types and operators. We currently employ the Vc library [6] for this specific purpose.

Starting from the implementation of the geometrical primitives available in the original USolids package, we have reengineered the code to make extensive use of generic programming patterns, possible in C++ with template classes and template functions. These template functions are used to compile into specialized binary code for the scalar interface, the vector interface, or the GPU kernel, as illustrated in Fig. 1.

![Diagram illustrating the software development approach in VecGeom. Templated C++ functions are written using abstract types and abstracted properties (traits). The different API then instantiates the template functions with concrete types and properties and the compiler produces different binary code for each case. In case of the vector API, SIMD vector types coming from the Vc library are injected.](image)

The primitives currently available in VecGeom v00.03.00 are: Box, Orb, Trapezoid (Trap), Simple Trapezoid (Trd), Sphere (+ spherical section), Tube (+ cylindrical section), Cut Tube, Cone (+ conical section), Generic Trapezoid (Arb8), Polycone, Polyhedron, Simple Extruded solid, Paraboloid, Parallelepiped (Para), Hyperboloid, Ellipsoid, Torus (+ toroidal section), Scaled and Boolean constructs. Vector signatures are implemented for each of them, allowing for concurrent queries.

The work made in refactoring the algorithms has provided the opportunity to introduce additional optimizations, like eliminating the use of some trigonometric functions often found in Geant4/ROOT implementations, or profit of template shape specializations for primitives that can assume different realizations, like, for example, the tube or cone shapes (hollow/full/cut variations).
Figure 2: Plot illustrating the performance figures (time in ms) achieved in VecGeom for the CutTube shape, for the various query functions, applied to the different backends and implementations.

3. CONCLUSIONS
The USolids primitives have been revised and extended to support vector signatures and are now part of the VecGeom package. Most of the shapes which are part of the standard set defining the GDML schema are now in place. The following shapes are either still missing or should be converted in VecGeom in order to complete the task: Tetrahedron (Tet), Multi-Union, Tessellated Solid, Generic Polycone, Extruded solid, Elliptical Cone, Elliptical Tube, Half-planes, Twisted shapes (box, trap, tube).
4. REFERENCES


ANNEX: GLOSSARY

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMD</td>
<td>Single Instruction, Multiple Data</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>GDML</td>
<td>Geometry Description Markup Language</td>
</tr>
<tr>
<td>USolids</td>
<td>Unified Solids</td>
</tr>
</tbody>
</table>