AVX2 Data Filtering Technique for Improved Vectorization

Guilherme Amadio

São Paulo State University

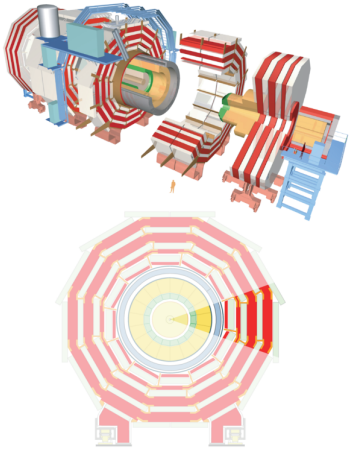
April 11, 2016

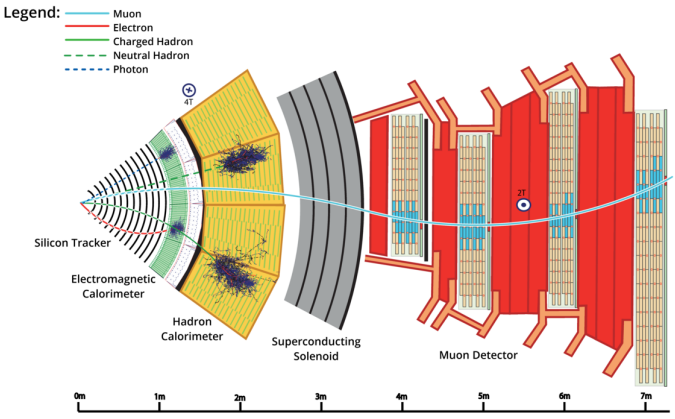
Overview

1. Background
   - Brief History of Geant
   - Overview of a Geant Simulation
   - Physics and Geometry Benchmarks
2. Data Filtering For Improved Vectorization
   - Motivation
   - Idea and Implementation
   - Proof of Concept

- Geant: GEometry ANd Tracking
  - Simulation Toolkit for the passage of particles through matter
- Geant History
  - First versions: 1974, 1976
  - Geant3: 1980, widely used by e.g. Fermilab
  - Geant4: 1994, move from Fortran to C++
- GeantV
  - Started by CERN/Fermilab
  - Project for modernization of Geant4
  - Complete rewrite of the whole code
  - International collaboration

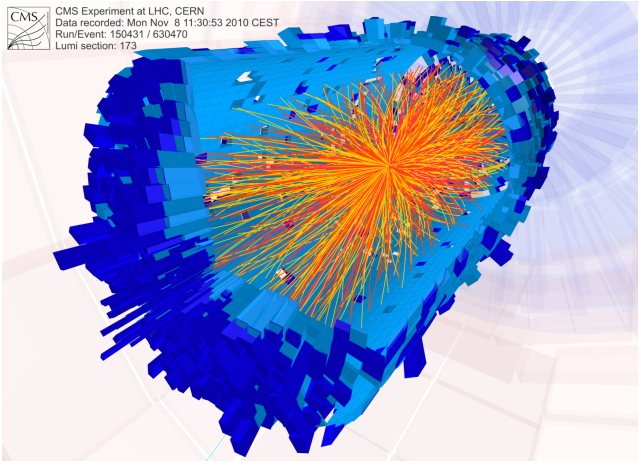
Overview of a Geant Simulation





Source: CMS Website — `https://cms.web.cern.ch`

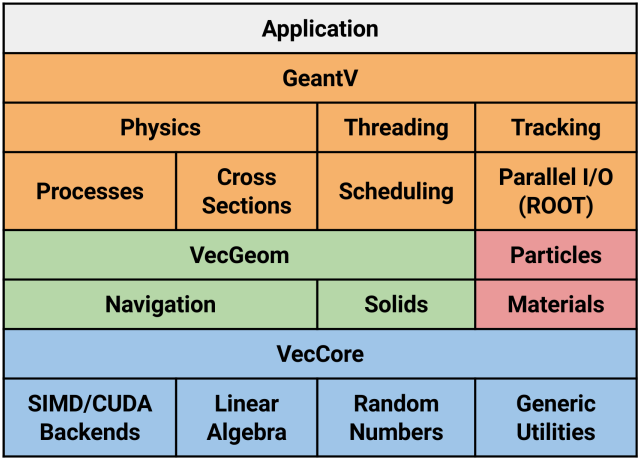
Visualization of a Collision Event



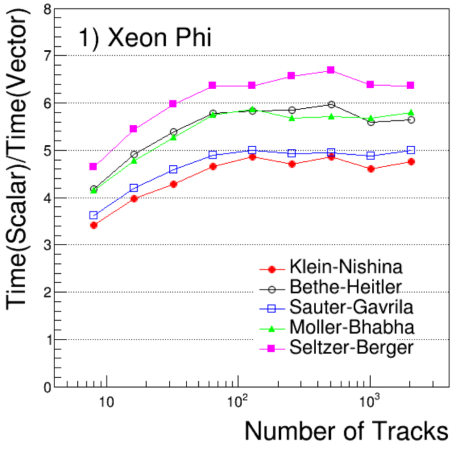
Source: `https://cds.cern.ch/record/1305179`

GeantV Simulation Toolkit

Project Structure

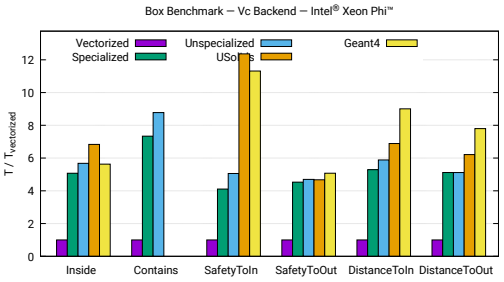


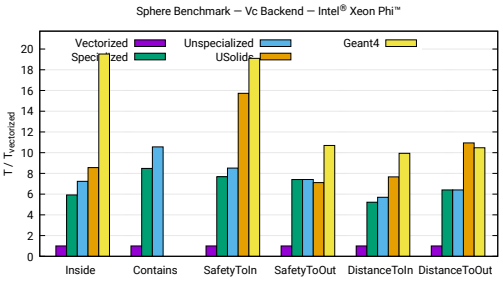
Performance: Electromagnetic Physics Models

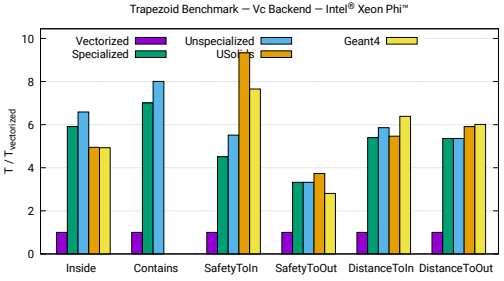


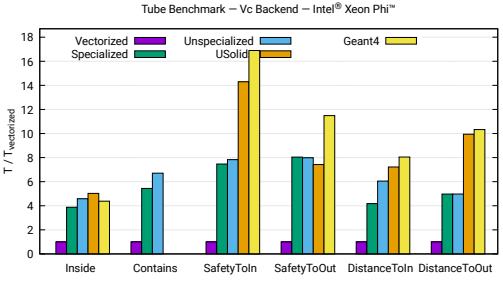
Note: Performance numbers shown here are preliminary

Performance: Geometry Algorithms









Motivation for Filtering Data

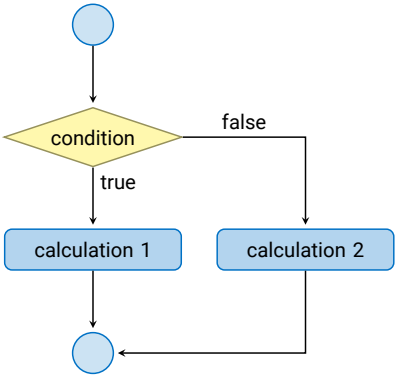
- Particles pass through different volumes within detector
- Particles undergo different physical processes depending on
  - Type (electron, positron, muon, proton, neutron, pion, neutrino)
  - Electric charge (neutral particles not affected by EM field)
  - Energy range (higher energies enable more processes, like pair production)
  - Material traversed (e.g., energy loss is higher in denser materials)
- New particles are created all the time, can reach 10^6 in single event
- Track baskets (similar to ray packets in ray tracing) can only take advantage of vectorization if they have similar characteristics
- Scheduler groups tracks into baskets according to geometry and physics
- Currently, only parts of geometry use vectorized algorithms

Motivation for Filtering Data (Cont.)

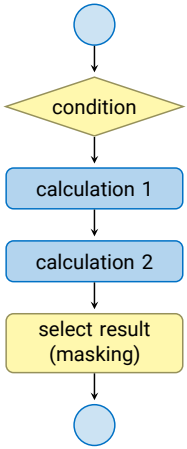
- Good vectorization gains in physics and geometry with vector API
- Need a strategy to process tracks via this API as much as possible
- Geometry is often cheap, but physics models are expensive
- Masking is not very effective, since branches are expensive
- Reorganizing data can be cheaper than computing both branches
- Therefore, filter data to group tracks to allow vector API to be used
- Data filtering using sorting networks on SIMD vectors with shuffle instructions
- Use track index as data to organize
- Gather groups of tracks such that all SIMD lanes are used

Branching is a Challenge for SIMD

Scalar operation

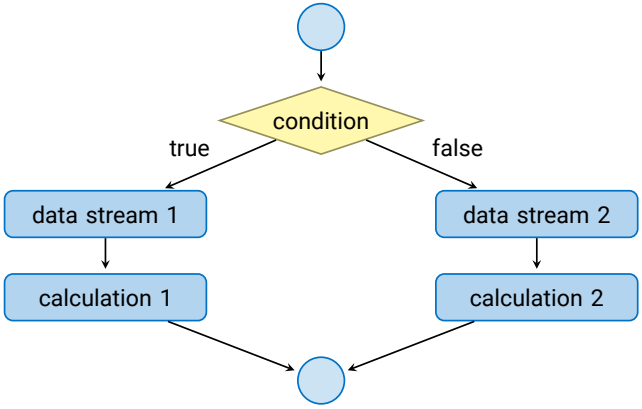


SIMD operation



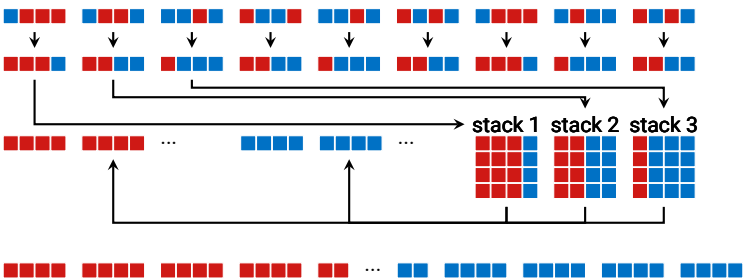
Sorting Network to Filter Data into Two Streams

SIMD Data Filtering



AVX2 Mask Sorting Network

- How can we avoid calculating both branches?
- Separate data into two streams
  - Sort within SIMD lanes with active/inactive mask
  - Use stacks according to number of active lanes
  - Transpose and pop full stacks to obtain separate streams



AVX2 Mask Sorting Network

- Sorting operations: swap inner/outer/mid pairs, cross product swizzle
- Variable `shtab` is a `true`/`false` table of 4 bits each
- Variable `code` contains the 4 bits for the current mask
- Using `vpermpd`, permute elements according to sorting network table

```
void _mm256_sort_pd(__m256d& mm, const uint8_t& mask)
{
    const uint64_t shtab = 0x311269308410200;
    uint8_t code = (uint8_t)((shtab >> 4*mask) & 0x0f);
    if (code == 0) return;
    if (code & 1) mm = _mm256_permute4x64_pd(mm, 0x4e);
    if (code & 2) mm = _mm256_permute4x64_pd(mm, 0xb1);
    if (code & 4) mm = _mm256_permute4x64_pd(mm, 0xd8);
    if (code & 8) mm = _mm256_permute4x64_pd(mm, 0x39);
}
```

source at: `https://github.com/IXPUG/WG_Vectorization`

Proof of Concept

Apply $f(x)$ or $g(x)$ if $x > 0.0$ for $x_i \in [-1, 1]$, on 1GB of **`double`**s where

$$f(x) = \frac{1}{1/x} \quad \text{and} \quad g(x) = f\left(\sqrt{x^2}\right).$$

The functions are just two artificially expensive forms of identity, to allow for easy inspection of the filtered result, and to create some work on each branch.
This technique applied to this case yields the results below (average of 5 runs).