Vectorizing Matrix Operations in the CMath Plugin

Beomki Yeo

Overview on Vectorization of Matrix Operations

- Why
  - Matrix operation used heavily in the track reconstruction (Runge-Kutta, covariance transport and KF)
- What
  - Vectorizing matrix operations in the **cmath** of algebra plugins
- How
  - **Auto-vectorization** rather than the explicit vectorization (**-march=native** compilation flag)
- In the very experimental stage
  - acts-project/algebra-plugins/pull/116
  - acts-project/detray/pull/717

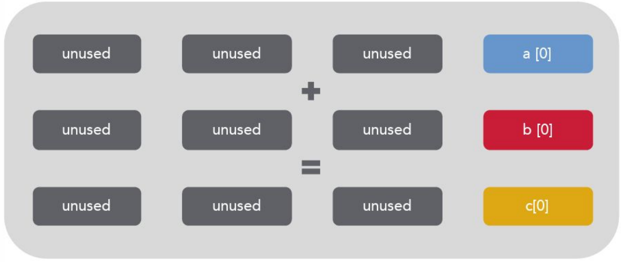
Vectorization

- The instruction on the multiple data of the register can be operated at the same time
    - SIMD
- There can be inefficiency in case the vector size does not fit into the register size or its integer multiplication

Example with four vector sum

```
for (i = 0; i < 4; i++)
  c[i] = a[i] + b[i];
```

Non-vectorized



Vectorized

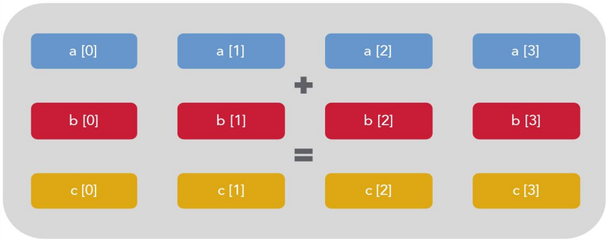
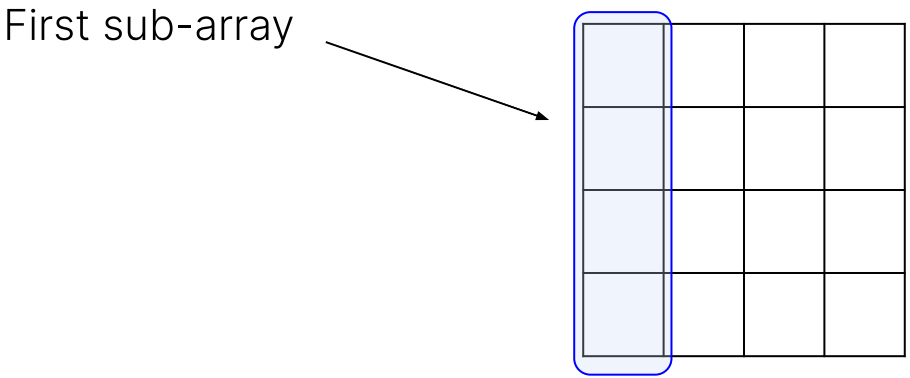


Image credit

Matrix Definition in the **cmath** plugin

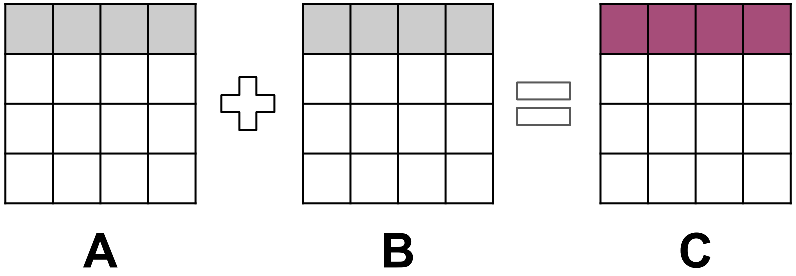
- Matrix is a 2 dimensional (jagged) array
  - Ex) 4×4 matrix = `std::array<std::array<float, 4>,4>`
  - Sub-array = column



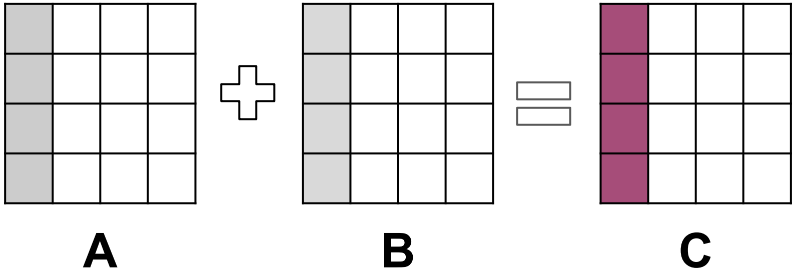
4 × 4 matrix

Vectorizing the Matrix Addition (A + B = C)

- Non-vectorized matrix addition for a row (**main**)



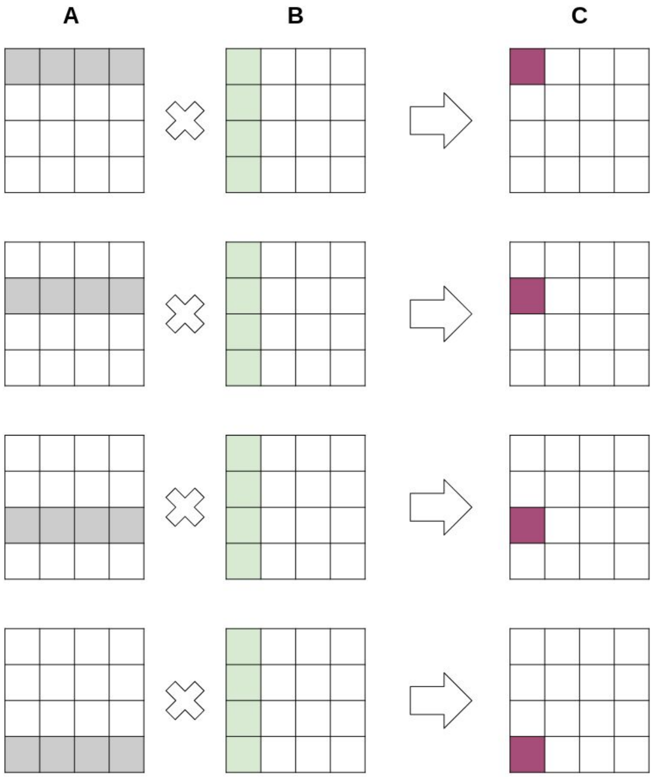
- Vectorized matrix addition for a column



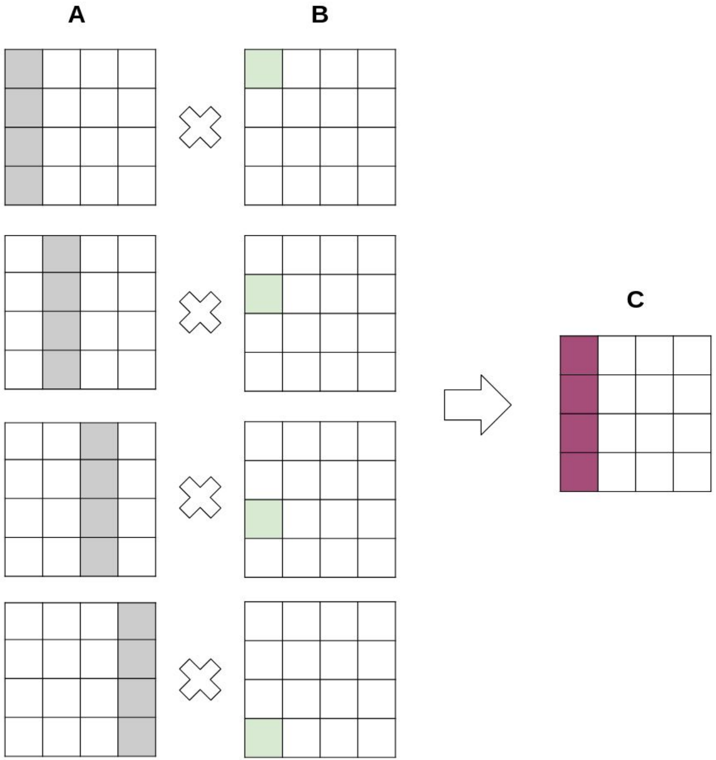

```diff
 template <typename size_type, template <typename, size_type> class array_t,
           typename scalar_t, size_type ROWS, size_type COLS,
           std::enable_if_t<std::is_scalar_v<scalar_t>, bool> = true>
 ALGEBRA_HOST_DEVICE inline array_t<array_t<scalar_t, ROWS>, COLS> operator+(
     const array_t<array_t<scalar_t, ROWS>, COLS> &A,
     const array_t<array_t<scalar_t, ROWS>, COLS> &B) {

   array_t<array_t<scalar_t, ROWS>, COLS> C;

-  for (size_type i = 0; i < ROWS; ++i) {
-    for (size_type j = 0; j < COLS; ++j) {
+  for (size_type j = 0; j < COLS; ++j) {
+    for (size_type i = 0; i < ROWS; ++i) {
       C[j][i] = A[j][i] + B[j][i];
     }
   }
```

Vectorizing Matrix Multiplication

- Non-vectorized matrix multiplication for a column

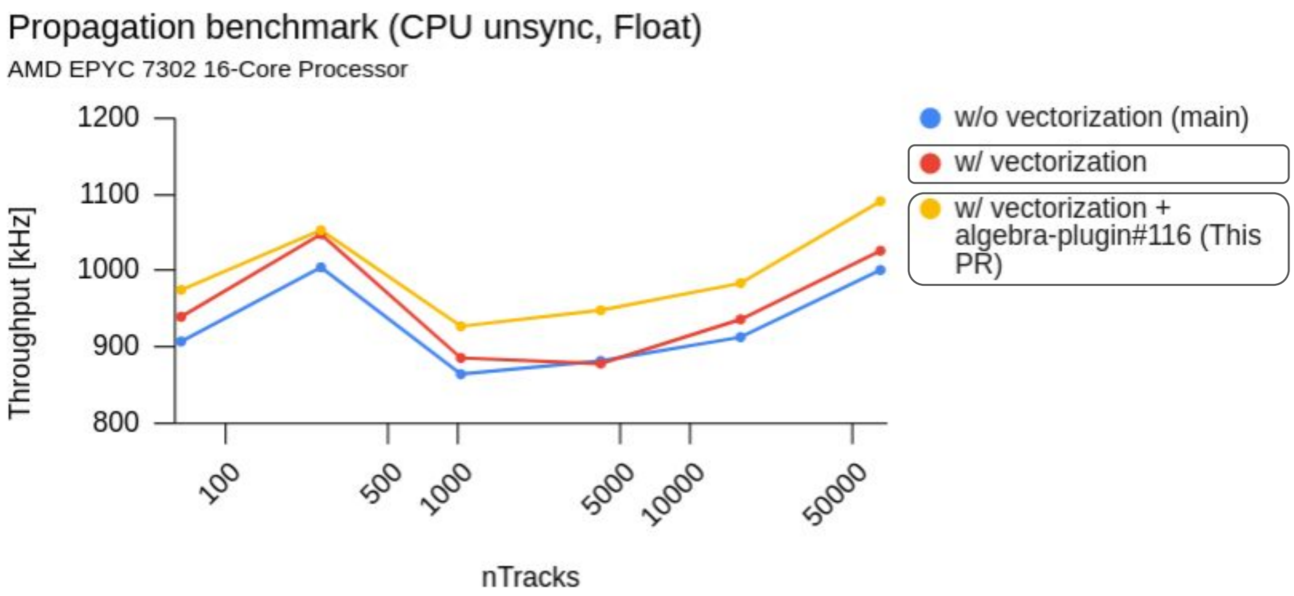


- Vectorized matrix multiplication for a column



Vectorization Performance (**Single**)

- Detray propagation benchmark in the toy geometry
  - Multi-thread with OpenMP
  - Includes the RK integration and covariance transport

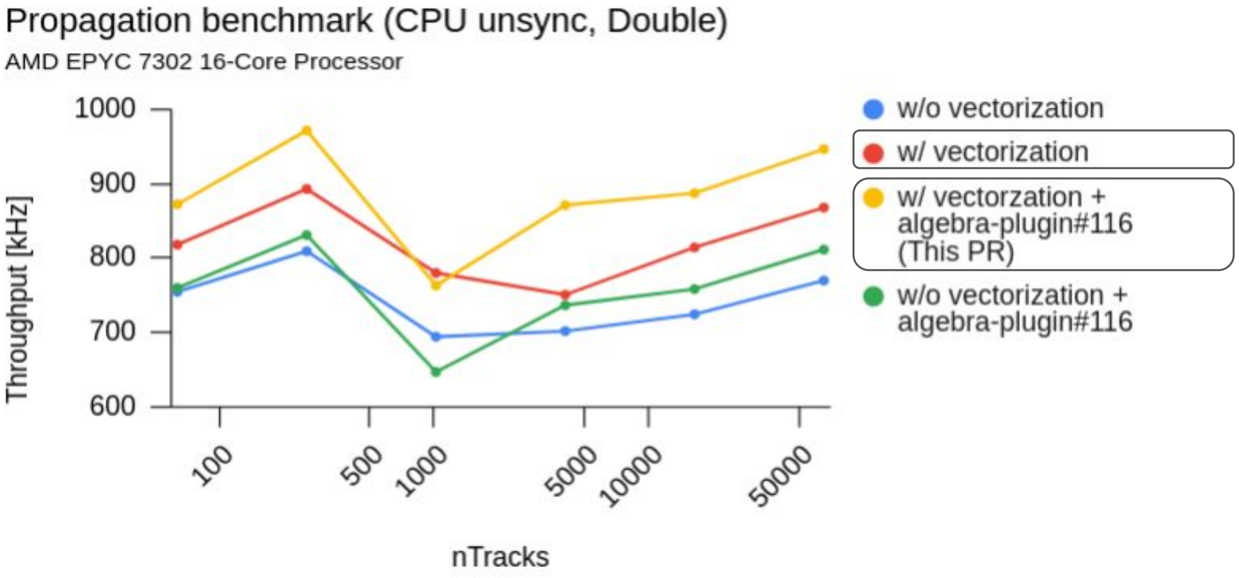


Only compilation flag

w/ Vectorizing matrix operations

Vectorization Performance (**Double**)

- Detray propagation benchmark in the toy geometry
  - Multi-thread with OpenMP
  - Includes the RK integration and covariance transport



Only compilation flag

w/ Vectorizing matrix operations

Prospect on the Matrix Inversion

- Only used for Kalman Filter (also CKF)
  - Have not investigated its impact, but it is good to optimize this as well

- Current matrix inversion algorithm is the *partial pivot LU decomposition*
  - Highly rely on **row-wise** gaussian elimination → Vectorization unfriendly
  - Should be replaced with **column-wise** gaussian elimination

Summary

- The vectorization on the matrix operation is tested (Very preliminary)
  - Performance seems to increase reasonably

- Matrix inversion should also be studied later