detray’s RKN Stepper compiler analysis

ACTS Parallelization Meeting
13.05.2022

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

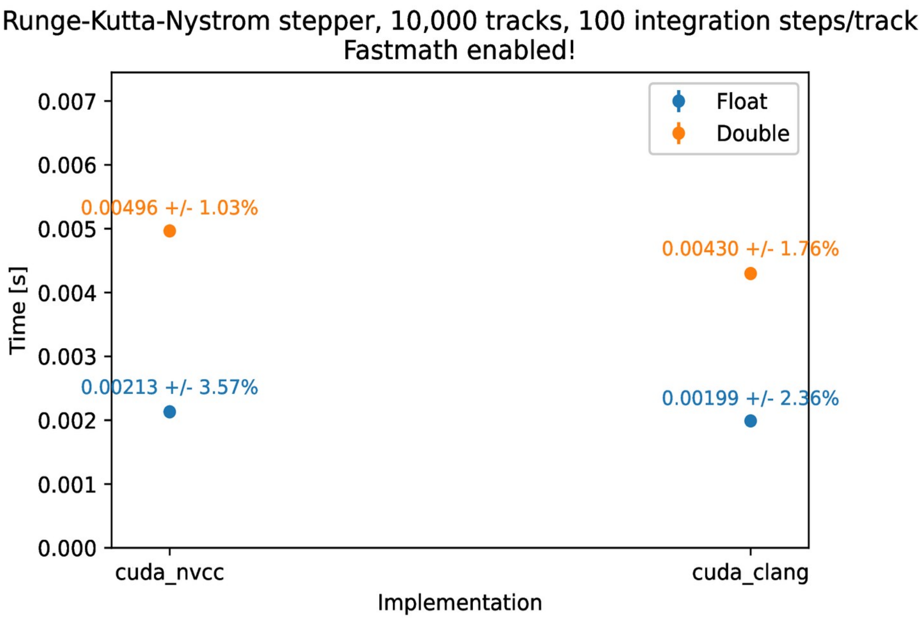
DESY.

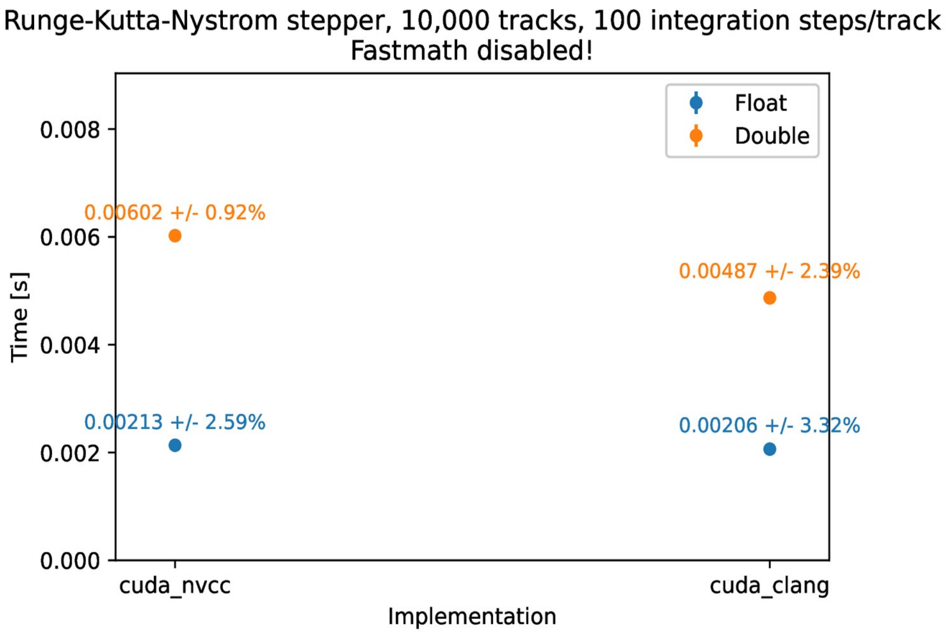
Setup

- Profiled code: detray/Runge-Kutta-Nystrom stepper CUDA kernel
  - [github/rk_stepper_cuda_kernel], code state: March 26th
- CUDA code compiled with
  - nvcc@11.6 + gcc@11.3 as host compiler : cuda_nvcc_gcc
  - nvcc@11.6 + clang@**13** as host compiler : cuda_nvcc_clang
  - clang@**14** only : cuda_clang
- CUDA driver version: 510.47.03

NVIDIA GeForce RTX 3060 (sm_86)

- detray backed = array
- DETRAY_CUSTOM_SCALARTYPE = float/double
- fastmath support = enabled/disabled





+15% (double)

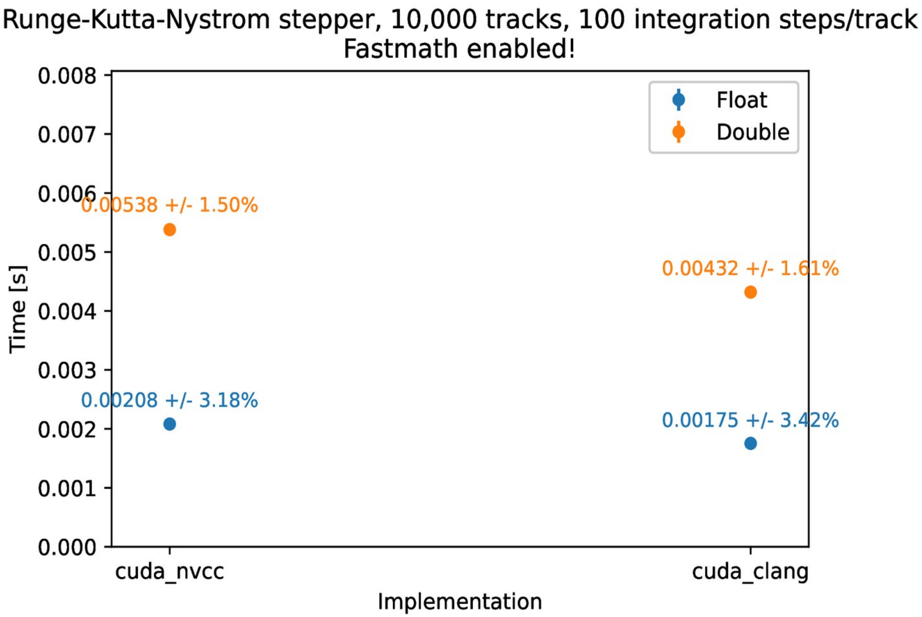
+7% (float)

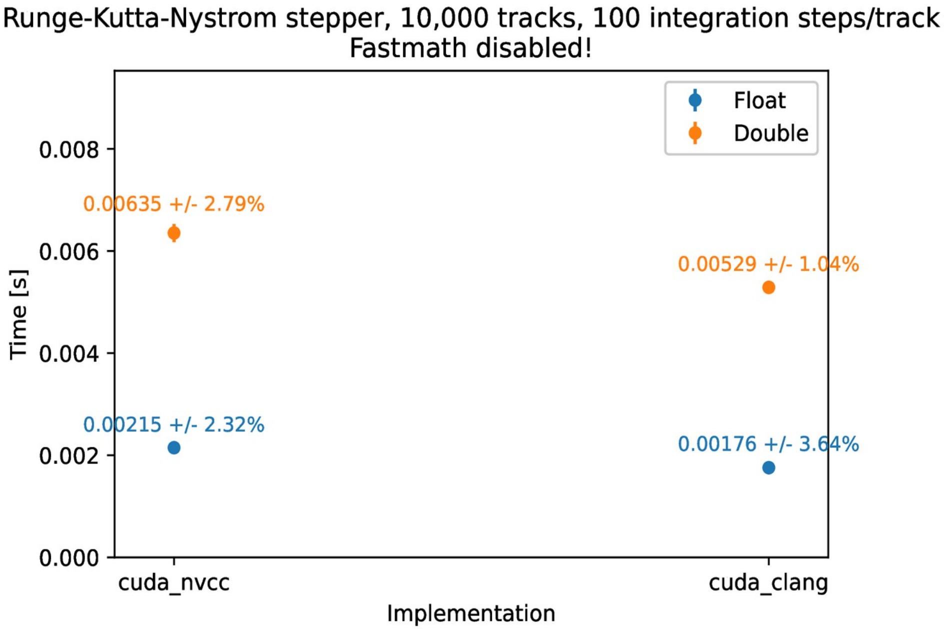
+23.7% (double)

+3% (float)

NVIDIA GeForce RTX 3060 (sm_86)

- detray backed = eigen
- DETRAY_CUSTOM_SCALARTYPE = float/double
- fastmath support = enabled/disabled





+24% (double)

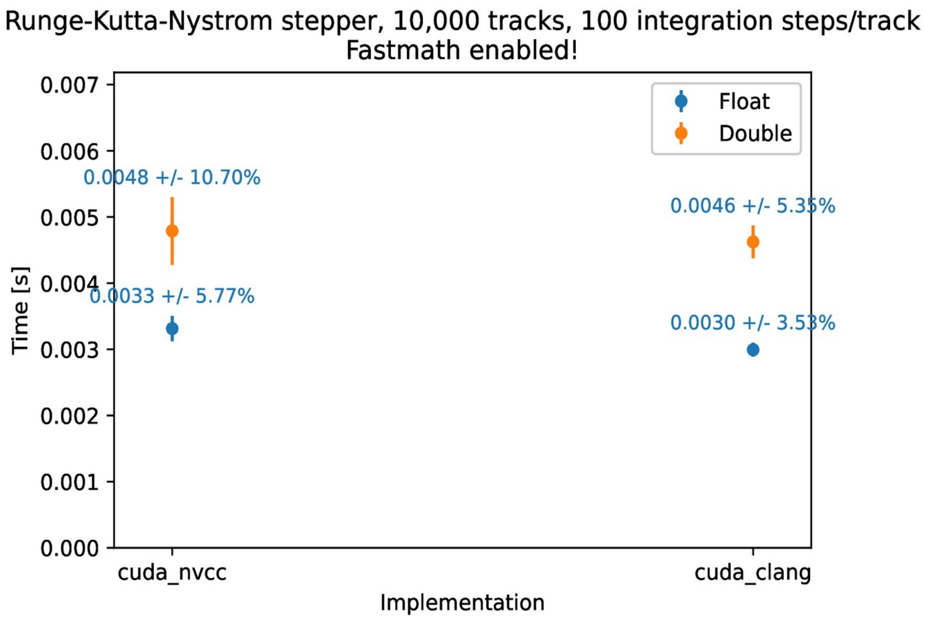
+18% (float)

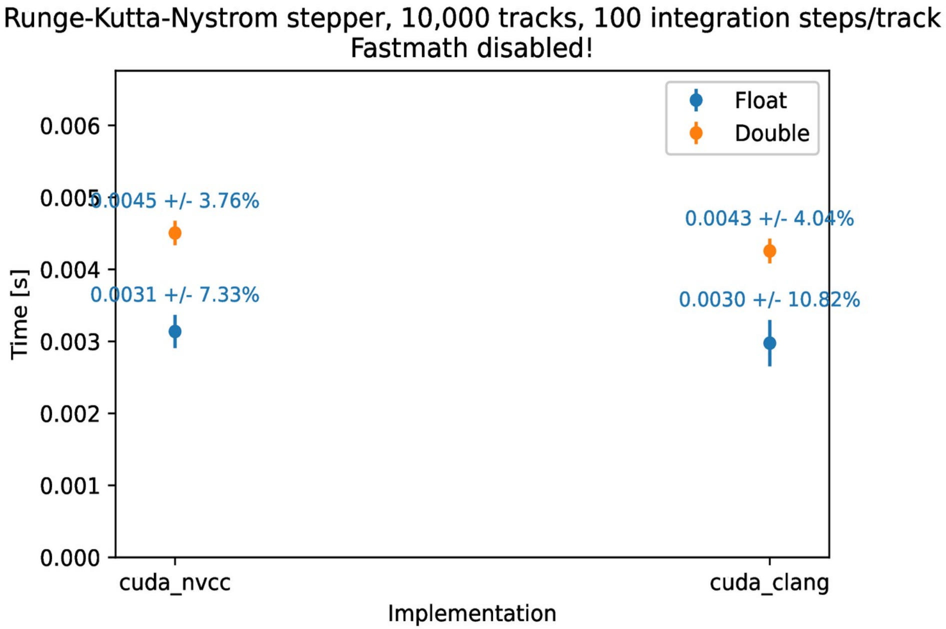
+20% (double)

+22% (float)

NVIDIA Tesla V100 (sm_70)

- detray backed = array
- DETRAY_CUSTOM_SCALARTYPE = float/double
- fastmath support = enabled/disabled





+3.5% (double)

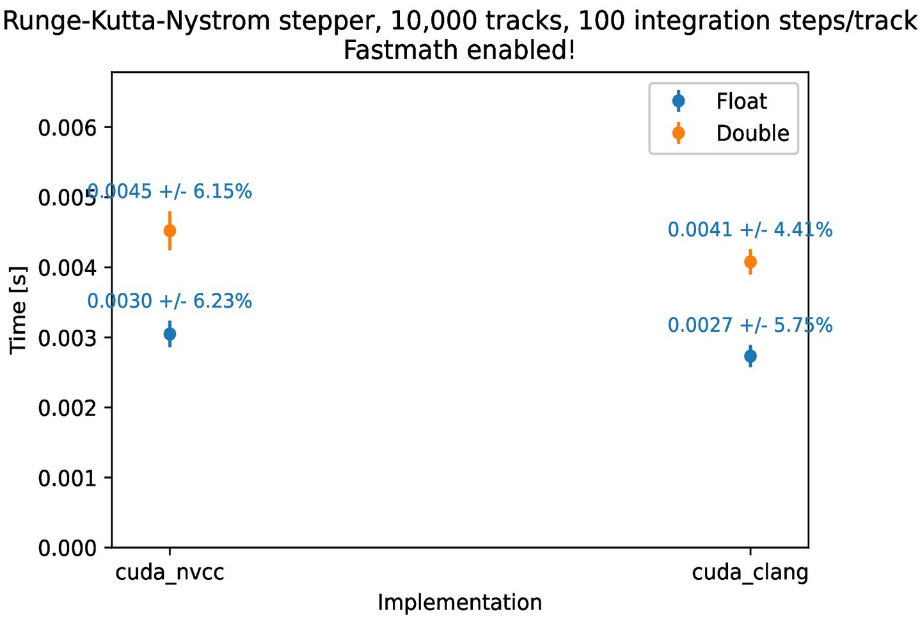
+10% (float)

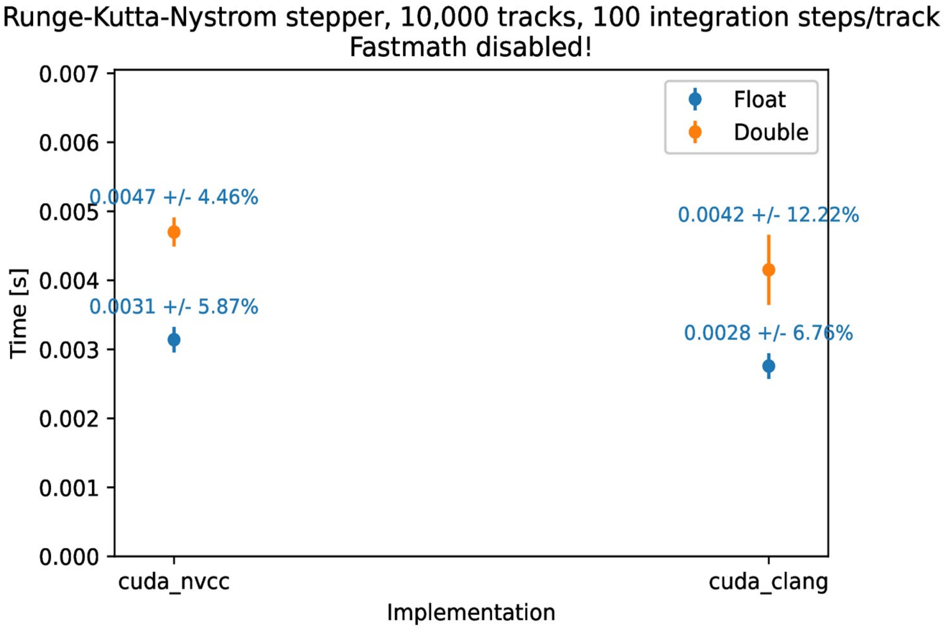
+5.8% (double)

+5.4% (float)

NVIDIA Tesla V100 (sm_70)

- detray backed = eigen
- DETRAY_CUSTOM_SCALARTYPE = float/double
- fastmath support = enabled/disabled





+10% (double)

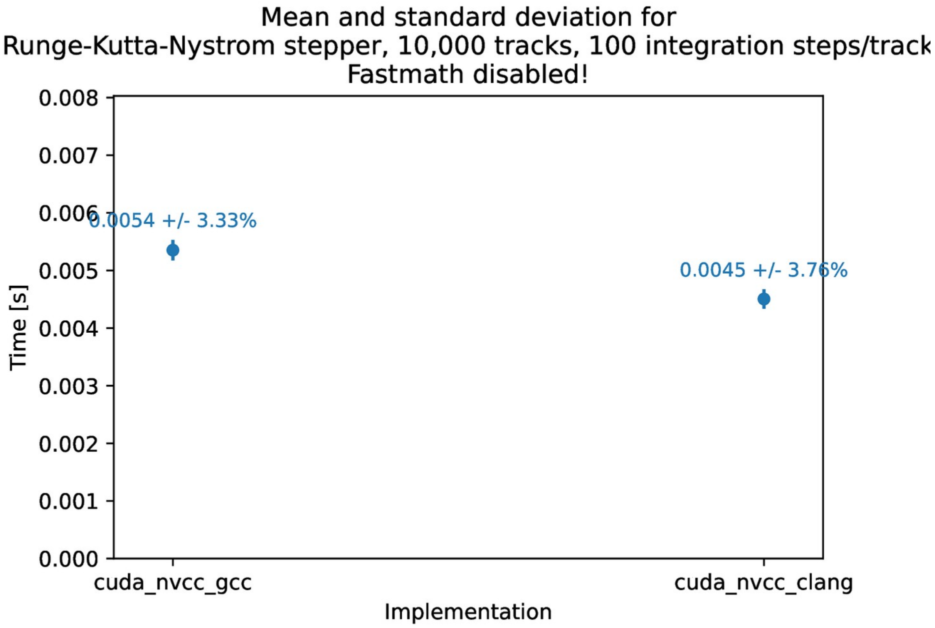
+11% (float)

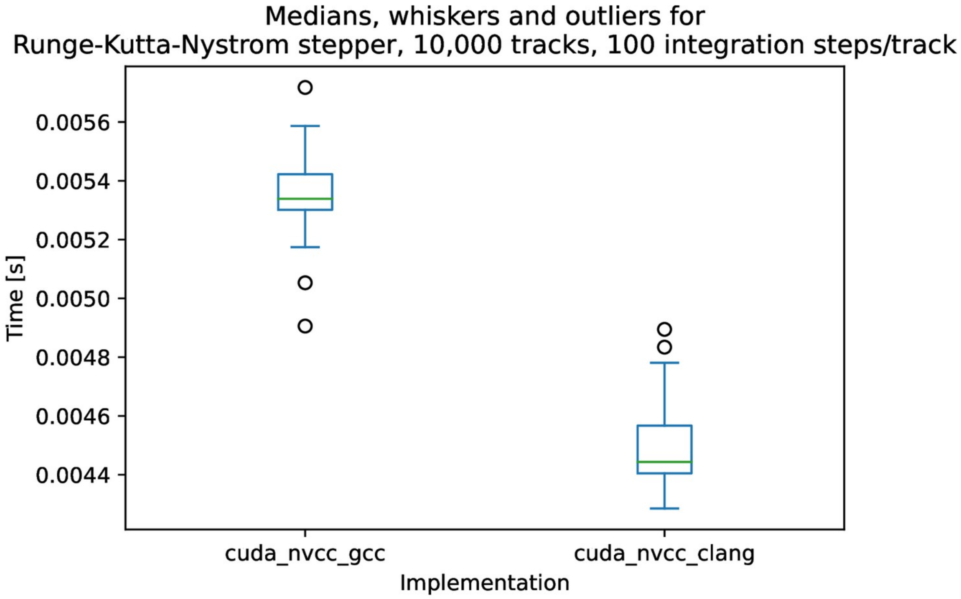
+13% (double)

+13.8% (float)

NVIDIA Tesla V100 (sm_70)

- detray backend = array
- DETRAY_CUSTOM_SCALARTYPE = double
- fastmath support = disabled





+18%

Observations

- An improvement in wall-clock time when compiling the CUDA code with clang instead of NVCC was observed in **all test cases** (both GPUs, in simple/double precision, with/without fastmath support, array/eigen backend)
- Potential gain is more significant when
  - double precision is used (up to 24%)
  - eigen backend is used over the array one
- nvcc with gcc as host compiler performs slightly worse than with clang
- Since the wall-clock time is in the order of milliseconds, the measurements could be influenced by operating system interactions. Would make sense to re-test these scenarios once more code is added to the kernel (e.g. the jacobian transport)