Celeritas: HEP detector simulation on GPUs Optimizing Dormand Prince method

Simon Barras, Julien Esseiva, Paolo Calafiura, Frédéric Bapst

University of applied science and architecture of Fribourg, Lawrence Berkeley National Laboratory

Celeritas

- Celeritas is a Monte Carlo particle transport code for simulating High Energy Physics detectors.
- GPUs accelerated.
- Motivated by the massive computational requirements of the High Luminosity LHC at CERN.
- Aimed to be used for physics simulation of the CMS and ATLAS detectors.
- It could be a plugin for GEANT4 or launch standalone.

CELERITAS

Dormand Prince

- Runge Kutta Dormand Prince (RKDP) is an Ordinary Differential Equations solver.
- Used to compute the path of particles.
- Particle state computed is a 3-dim vector of the position and another one for the momentum.

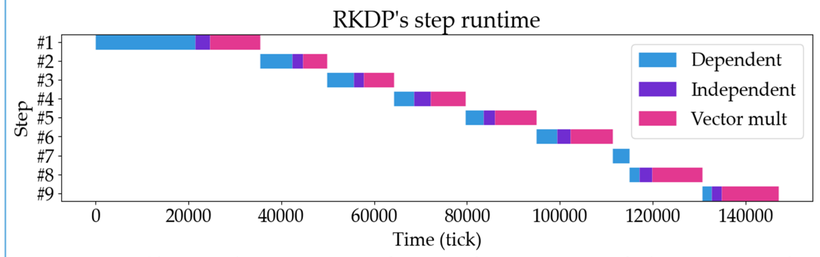

Figure 3 – Shows the runtime of a single iteration of the RKDP. The dependent part of the runtime is mostly the execution of the equation. The vector multiplication is the update of the particle's state. Independent part is the compute of the coefficients defined by the butcher table and multiplied by the step size.

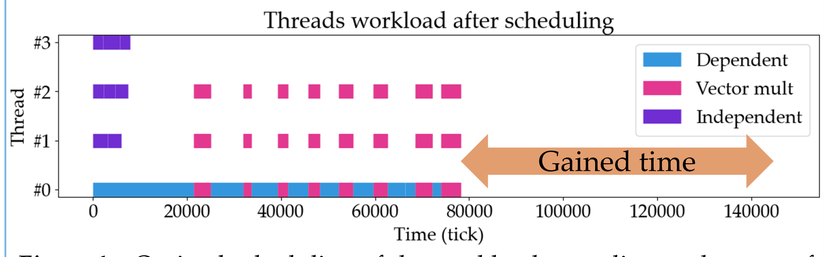

Figure 4 – Optimal scheduling of the workload according to the type of the task without the time spent in thread synchronization and message passing.

When using four threads to compute one iteration of RKDP and neglect thread synchronization, the runtime can be divided by a factor up to two. The time Before realizing the new implementation, the real gained time cannot be known.

Motivation

- Runge Kutta Dormand Prince (RKDP) is a crucial kernel of the particle propagation action, Celeritas' most time-consuming.
- Optimizing RKDP offers large potential payoff.

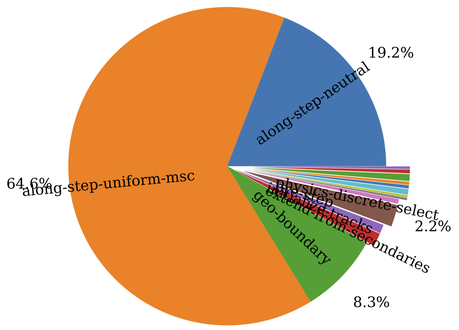

Figure 1 – Percent of the runtime per action. (source: Julien Esseiva).

GPU Programming

- Single Instruction Multiple Data (SIMD)
- High number of cores

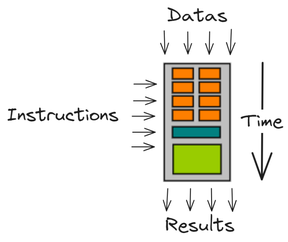

Figure 5 – Schema of a GPU's microprocessor. To deal with SIMD, the GPUs use multiple microprocessors that execute physically multiple threads.

According to the situation on the figure 4, a thread should have the role of a coordinator. This is not a common way to use the GPUs, but the CUDA interface provides a set of instructions called "warp shuffle" that allow us to synchronize groups of 32 threads (called "warps") and exchange information between them.

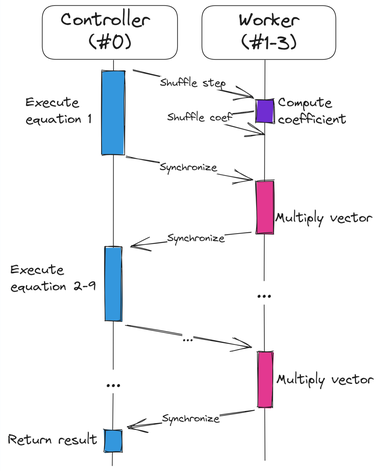

Figure 6 – Sequence diagram illustrating the synchronization between the threads. The right column represents 3 threads. The color of each task refers to the type explained in the figure 3 and 4.

Along Step Uniform MSC

- Multiple Scattering (MSC)
- Executed for every track
- One GPU thread per track

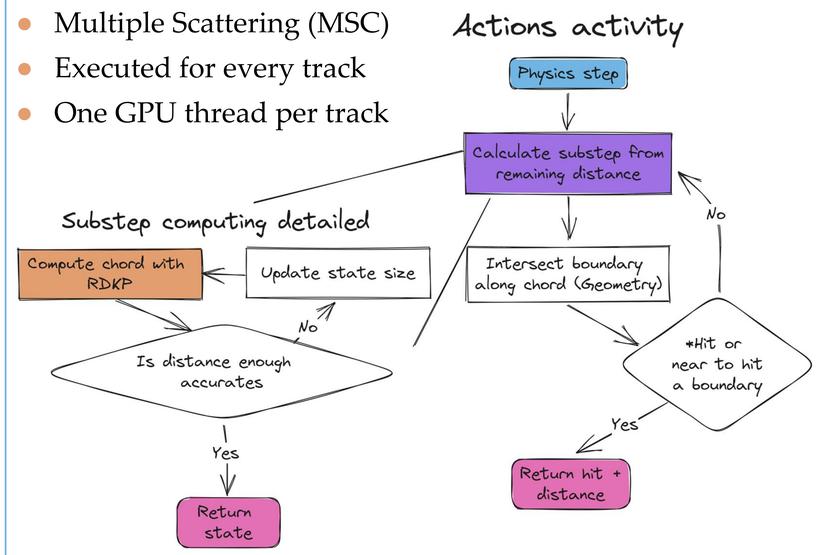

Figure 2 – Activity diagram inspired by the presentation made by Seth Johnson at the CMS update 2019. * The condition and the actions to finish the Along Step Uniform MSC flow is highly simplified in this schema. The usage of RDKP could be used with different tolerances through the simulation if the result are hard to find.

The RDKP method is executed at least one time every time a chord must be computed. A new chord is computed until the track hit a boundary. According to the data resulted during a CMS experiment, RDKP is mostly executed only one time to give an acceptable state.

Acknowledgements

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and Office of High Energy Physics, Scientific Discovery through Advanced Computing (SciDAC) program.

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725