Submission Details: pos115s1

Form first submitted: 2024-12-13 08:00 CST
Form last updated: 2025-04-04 10:33 CDT

Scientific Domain

Engineering: 3
Physics: 2
Computational Methods and Applied Mathematics: 1

Title (Maximum 25 words)

Title (Maximum 25 words): Graph Abstraction for Efficient Scheduling of Synchronous Workloads on GPU

Author Information

Author 1:
Gender: male
Name: Mr. Romin Tomasetti
Email: romin.tomasetti@gmail.com
Affiliation: University of Liège
2nd Affiliation:
Country of the Primary Affiliation: Belgium
Will this person present the poster at the conference? Yes

Author 2:
Gender: male
Name: Prof. Maarten Arnst
Email: maarten.arnst@uliege.be
Affiliation: University of Liège
2nd Affiliation:
Country of the Primary Affiliation: Belgium
Will this person present the poster at the conference? No

Long Abstract

Long Abstract (Maximum 800 words):

Many computational physics simulations need to efficiently execute asynchronous workloads (FEM assembly, linear algebra, etc) that can be organised as a Direct Acyclic Graph (DAG).

Ad hoc scheduling of these asynchronous workloads is an additional burden to the code and might not fully exploit the available execution resources (e.g. a multi-GPU node). By contrast, architecting the code based on a graph abstraction exposes the whole computational graph to the compiler/driver
ahead of execution, thereby enabling as many optimisations as possible. Therefore, a graph abstraction that can be prescribed either at compile time or at runtime is necessary, and it must be mappable to the best backend scheduler, thus maximising resource usage.

We contribute to the *Kokkos* implementation of this graph abstraction which allows for a performance portable single source code. More specifically, this poster will focus on recent contributions to *Kokkos::Graph* that make it evolve towards the C++ *std::execution* proposal for managing

asynchronous execution on generic execution resources (P2300).

We will demonstrate the benefits of using *Kokkos::Graph* both in terms of performance and software design. We will present several examples of varying complexity, including a FEM simulation of electromagnetic wave scattering.

Short Abstract

Short Abstract (Maximum 200 words):

Many computational physics simulations need to efficiently execute asynchronous workloads (FEM assembly, linear algebra, etc) that can be organised as a Direct Acyclic Graph (DAG). Ad hoc scheduling of these asynchronous workloads is an additional burden to the code and might not fully exploit the available execution resources (e.g. a multi-GPU node). By contrast, architecting the code based on a graph abstraction exposes the whole computational graph to the compiler/driver ahead of execution, thereby enabling as many optimisations as possible. Therefore, a graph abstraction that can be prescribed either at compile time or at runtime is necessary, and it must be mappable to the best backend scheduler, thus maximising resource usage. We contribute to the Kokkos implementation of this graph abstraction which allows for a performance portable single source code. More specifically, this poster will focus on recent contributions to Kokkos::Graph that make it evolve towards the C++ std::execution proposal for managing asynchronous execution on generic execution resources (P2300). We will demonstrate the benefits of using Kokkos::Graph both in terms of performance and software design. We will present several examples of varying complexity, including a FEM simulation of electromagnetic wave scattering.

Acknowledgement

Acknowledgement: yes