Implementation of high-performance GPU kernels in *Nektar++*

Supporting: Jan Eichstädt (3 months)
Collaborators: Dr. David Moxey, (Engineering, University of Exeter), Prof. Spencer J. Sherwin (Aeronautics, Imperial College London), Prof. Joaquim Peiró (Aeronautics, Imperial College London)

1 Outline

In modern high-performance computer architectures, both CPUs and GPUs, the gap between processor clock speed and memory bandwidth has seen a consistent increase over the last decade. In this context, high-order finite element methods are particularly attractive due to their high arithmetic intensity, which is further tuneable through the choice of element polynomial order. Matrix-free formulations of operators, on tensor product elements, in combination with single instruction multiple data (SIMD) vectorisation is a well studied solution for efficient implementations. Recently, the above strategy has been extended by Moxey *et al.* [1] to tensor product simplicial elements on a standalone app, which is partially based on the *Nektar++* library, for the solution of the Helmholtz equation. The *Nektar++* library is a very suitable environment for such efforts, as it is a tensor product based *hp*-spectral element framework that comes with a number of PDE solvers, and also allows one to construct a variety of new solvers. The main existing solvers are a continuous Galerkin incompressible Navier-Stokes solver and a discontinuous Galerkin compressible Navier-Stokes solver.

I am currently extending the above mentioned standalone app with efficient GPU kernels for the Helmholtz solver, which come in two flavours: Firstly SIMT vectorised kernels similar to the CPU kernels, and secondly non-vectorised kernels which exploit the inner parallelism of each elemental operation. In this project we plan to port the core GPU kernels of this app, thus adding a GPU backend to the *Nektar++* library. In a previous project, G. Castiglioni has already started porting the vectorised CPU kernels to a new library, so this project will build on his efforts. The intent is to improve the efficiency of key operators of the *Nektar++* library with the end goal of accelerating both the compressible and incompressible flow solvers. In effect, this project will lay the foundations for the first GPU implementations of the *Nektar++* library. So far the GPU kernels have been implemented using the *CUDA* programming model for Nvidia GPUs, but in order to also address AMD GPUs, that are increasingly installed on HPC clusters, we will additionally look into the *HIP* [2] programming model, which is syntactically very similar to *CUDA* . The purpose of this project is to apply techniques to accelerate high-order finite element operators which will not only benefit the *Nektar++* user base, but also both other PRISM partners and the wider academic community.

2 Project objectives

The end goal of this project is to improve the computational efficiency of key operators within the *Nektar++* library to exploit GPU hardware.

Porting GPU kernels: As first step towards the end goal, I will port GPU kernels for matrix-free operators from an existing app [1] to *Nektar++* to add to the new core library. The app was developed to showcase the advantages of matrix-free operators for tensor product simplicial elements in the context of a Helmholtz solver. Most of these operators are the backbone for more complete Navier-Stokes solvers. The app further served as a *sandbox* to investigate

different parallelisation and memory placement approaches for efficient GPU kernels. Now that we have distilled the most performant GPU kernels, the aim is to backport these to the *Nektar++* library. This is facilitated as the app already utilises the *Nektar++* library for the construction of basis functions, derivatives, quadrature points, weights, and other ancillary functions. At this point it will also be essential to decide on the specific GPU programming model, either *CUDA* or *HIP*.

Developing missing kernels: Not all kernels for all element types were developed for the app: the second step will consist in developing the missing kernels.

Benchmarking kernels: The third step of this project will consist of benchmarking the newly ported kernels against the currently used ones. The benchmarking will cover 2D and 3D elements such as quadrilaterals and triangles in 2D hexahedrals, prisms, and tetrahedrals in 3D. All elements will be tested in in regular and deformed configurations. The benchmark metrics will comprise througput/DOF, and roofline models based on different memory and cache level bandwidths.

Publicating the results: The parallelisation approaches (SIMT vectorisation vs inner parallelism), as well as data placement strategies for different GPU memory spaces shall be presented in a scientific publication. This will entail the presentation of the benchmarking results mentioned above.

3 Alignment with PRISM strategy

Supporting long-term research: This project will allow me to spend the time necessary to backport extensive parts of my research to the *Nektar++* library, therefore directly benefit the *Nektar++* group and the wider *Nektar++* community. The solver restructuring will also create the foundation for the first GPU implementations of *Nektar++* flow solvers. Given the increasing prevalance of GPUs in leading HPC clusters, this step will increase the chances of creating successful proposals for runtimes on these machines.

Development of *Nektar++* developer: Additionally, this fund will enable me to focus on integrating major parts of a library to an extensive framework such as *Nektar++* , therefore strengthening my programming skillset and thus becoming a more valuable computational researcher.

Collaboration with other PRISM members: Dr. Peter Vincent is another PRISM investigator and he is a project leader of *PyFR*. *PyFR* is a solver based on flux reconstruction which can be seen as another flavor of high-order finite element methods. The techniques parallelisation approaches, memory placement, and the comparison of *CUDA* and *HIP* programming model, could potentially benefit developments in *PyFR*.

4 Brief work plan

- Port GPU kernels for the Helmholtz solver that were developed by myself to the new core library in *Nektar++* .
- Develop missing kernels for all element types.
- Benchmark newly ported kernels against vectorised CPU kernels.
- Write and submit a journal publication to disseminate the results to the wider academic community.

References

[1] D. Moxey, R. Amici, and R. M. Kirby. Efficient matrix-free high-order finite element evaluation for simplicial elements. under review in SIAM J. Sci. Comput., March 2019.

[2] HIP Repository. https://github.com/ROCm-Developer-Tools/HIP, 2020.