Accelerating Performance of NekCEM with MPI and CUDA

Azamat Mametjanov, Misun Min, Boyana Norris and Paul Hovland
Mathematics and Computer Science Division
Argonne National Laboratory

Abstract

Modern supercomputing architectures are increasing the number of cores per node by adding GPU co-processors to increase instruction and memory throughput. Compute-intensive applications need to take advantage of higher throughput by employing both distributed and shared memory programming models. In this paper, we describe our approach for accelerating Computational ElectroMagnetics application NekCEM using MPI and CUDA programming models and demonstrate its performance on the leadership-class computing systems. The resulting code kernels can potentially be reused in other applications within the Nek family of codes.

1 Introduction

NekCEM is an open-source, scalable implementation of high-order methods for electromagnetic (EM) device simulations [1]. One of the primary applications is a simulation of an undulator – a high-energy EM device that bends electrons forcing them to radiate intense and concentrated energy for use in particle accelerators and colliders. By enabling the simulation of EM dynamics, NekCEM complements and strengthens the theoretical and experimental research in high-energy physics.

NekCEM uses an MPI-based single-program multiple-data (SPMD) programming model, where each MPI process executes on its piece of a decomposed domain and cooperates with neighboring processes by exchanging messages for process boundary data points. The calculations are done using spectral element discontinuous Galerkin (SEDG) scheme, where electric and magnetic field values on just the domain boundary faces are exchanged between neighbors. Upon completion of face data exchange, a new time-step calculation is initiated with the new data from neighbors.

Due to the relatively small data exchange, the execution time of a simulation run is computationally bound such that for most runs 25 to 50% of overall time is spent in a matrix-matrix multiplication routine (`mxm`). This is due to the computational intensity of looping through each mesh element and calculating the local gradient along each dimension of the modeled domain. Therefore, it is essential to accelerate the execution of `mxm` operation to improve the overall performance of application runs.

In this paper, our contributions are as follows

1. We describe the parallelization of `mxm` routine using CUDA threads
2. Due to the inherent characteristics of a GPU as a co-processor with its own memory hierarchy, it is insufficient to simply port the code for execution on the GPU. We describe several optimizations to obtain additional performance improvements.
3. Finally, acceleration of NekCEM is intended to improve other codes in the Nek-family of applications that use a similar computational core.

2 Background

In this section, we describe the existing structure of NekCEM to observe performance bottlenecks and opportunities for acceleration.

An application run consists of three stages: setup, solve and checkpoint stages. The setup stage initializes processors, reads and distributes mesh data to the processors, and assigns geometric coordinates to mesh points. The solve stage performs time-stepping iterations and evaluates spatial operators at each time step. The checkpoint stage produces output files and checkpoints global field data.

The solver stage invokes a 5-stage fourth order Runge-Kutta time-stepping routine, which was previously shown to produce favorable results in comparison with those from low-order methods [2][1]. In this routine, the spatial Maxwell operator is invoked five times with the corresponding pre- and post-processing of operator results. The operator itself computes field values within a processor's domain, computes face values and exchanges them with neighboring processors.

The most computationally intensive routine within the operator is the calculation of weighted curl values. Here, for each EM field along mesh dimensions (e.g. $E_x, E_y, E_z, H_x, H_y, H_z$ in 3-D) and for each mesh element e, the routine `local_grad3` is invoked (see the following figure).

3 Approach

4 Analysis

5 Related Work

6 Conclusion

Acknowledgments

This work was supported by the U.S. Department of Energy Office of Science under Contract No. DE-AC02-06CH11357.

References

[1] Misun Min and Paul Fischer. Spectral-element discontinuous galerkin simulations with a moving window algorithm for wakefield calculations. In *Proceedings of Particle Accelerator Conference (PAC'09)*, 2009.

[2] Misun Min, Paul Fischer, and Y. C. Chae. Wake fields for tesla cavity structures: Spectral element discontinuous galerkin simulations. In *Proceedings of SRF (SRF'07)*, 2009.