Computer Science Clinic

Midyear Update for
Sandia National Laboratories

Parallelizing Intrepid with Kokkos

December 7, 2014

Team Members
Alex Gruver
Ellen Hui
Tyler Marklyn
Brett Collins (Project Manager)

Advisor
Jeff Amelang

Liaison
Carter Edwards

Progress

Our main objective for this semester was to rewrite the tensor contraction functions of Trilinos' Intrepid package to be thread scalable. To do this, we used Kokkos - a library developed by Sandia to allow simultaneous programming for the GPU and CPU.

We began the semester by working through a short online class as well as many example problems to sharpen our GPU programming skills. Optimizing code for GPU performance is a career, but by tackling example problems before moving on to Intrepid we were able to familiarize ourselves with GPU programming enough to make meaningful contributions to the code.

Since then, we have been able to export multiple kernels from Intrepid and get them working with Kokkos. We were hoping to achieve speedup on the order of 100x, but since the original intrepid kernels were implemented in a cache-friendly way, we have changed our expectations to 20-40x speedup for kernels suited to GPU programming. On one of the kernels, contractFieldFieldScalar, we have been able to accomplish a speedup of 35x, using methods that are reproducible in Kokkos.

During this time we have learned a lot about using Kokkos, which was one of the most important aspects of the project, as well as GPU programming in general. We are now at the point where we understand how to create faster Kokkos versions of the contraction kernels in Intrepid. However, we have observed some disparity in the performances of Cuda and Kokkos. Hopefully we'll be able to resolve these differences going forward.

Obstacles

Initially, we planned on completing the fully re-engineered Intrepid package using Kokkos by early December. While we were not able to finish parallelizing the entire Intrepid package with sufficient speedup, we plan on delivering a subset of the kernels by winter break, and giving a seminar on how to best achieve speedup using Kokkos at our site visit.

Our main obstacle this semester has been the difficulty of reasoning about memory access patterns on the GPU. Most of our time has been spent determining general memory access patterns that can be applied to many of the kernels in the package.

Another obstacle we've encountered is the low ratio of work to memory accesses inherent in the nature of a few kernels. The kernel that performs

dot products, for example, has to perform one memory access per multiply. Because memory accesses are so expensive on the GPU, it is incredibly difficult to get significant speedup on problems like these.

At this point, we have couple prototype kernels producing performance gains on the expected order of magnitude. Given the amount of time left in the semester, it is likely that we will have to settle for presenting a subset of the kernels instead of a fully reworked Intrepid package in December.

Future Plans

Finishing the Kokkos versions of Intrepid's ArrayTools functions is our prority going into the spring. We were originally hoping to finish this task by Winter Break (December 16th), but we have come across unexpected problems as mentioned in the Obstacles section. We plan on focusing our efforts on a couple of the simpler methods in the ArrayTools library in order to get the desired speedup. It is more important for this clinic project to demonstrate the capabilites of Kokkos than it is for it to overhaul the entire codebase of Intrepid. Furthermore, we believe that once we have correct Kokkos implementations for a couple of the functions we can apply similar algorithmic techniques to speed up the remaining functions in Intrepid.

While working on this project, we have and will continue to record our thoughts about Kokkos in a location accessible by our liasons. These notes will give Sandia valuable feedback about Kokkos and information on how to improve the user experience. Logging these notes is another of our priorities, since we are currently serving as Kokkos beta testers.

Depending on the upcoming work and whether we have a breakthrough in speedup, we may meet our Winter Break deadline or finish with Intrepid early next semester. We do not yet know the specifics of our next task, but it will most likely be creating a Kokkos version of a different high performance computing package. Our approach to this next task will be similar to the one we used on Intrepid. We will begin by finding the simplest functions in the package, then creating Kokkos versions of those functions and manipulating memory accesses and parallelization points until we achieve reasonable speedup. Then we will apply the algorithm to the rest of the functions. Our goal is to finish the Intrepid package as well as at least one more package by the end of the school year.

Our stretch goals are creating Kokkos versions of more high performance computing kernels and integrating the Kokkos versions of the kernels back into the original packages. We are leaning towards creating Kokkos

versions of more kernels because we will understand Kokkos and what is required to gain speedup. Integrating the Kokkos versions of the kernels into their original packages would bring up new challenges that we have not yet faced and therefore would have a much greater learning curve.

Overall, our priorities, in order, are:

- Log our experiences with Kokkos on the team wiki
- Finish creating a Kokkos version of Intrepid
- Report on our methodology for writing Kokkos versions of Intrepid kernels
- Repeat this process with a new suite of high performance computing kernels

We should certainly be able to get the first three of these tasks done by the end of the clinic project, with the fourth being a reachable stretch goal.