Computer Science Clinic

Midyear Update for
Sandia National Laboratories

Parallelizing Intrepid with Kokkos

December 4, 2014

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

Advisor
Jeff Amelang

Liaison
Carter Edwards

Progress

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. Thus far, we have not been able to get as much speed-up as we originally expected. However, 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. Additionally, we do see speed-up on these kernels for Sandia's use case, which involves repeated application of the kernels to data stored on the GPU. This is a boon for us in terms of speed-up because it allows us to avoid expensive data copies from the GPU to the CPU.

Obstacles

Our initial plan for this semester called for a fully re-engineered Intrepid package using Kokkos by early December, which we would present in a seminar during our site visit to Sandia at the end of the semester. However, we have run into a few setbacks, and are therefore unlikely to meet that goal.

Originally, we had hoped to find a working algorithm using Kokkos that yielded speed-up comparable to hand-coded Cuda implementations of matrix multiplication. However, our prototype Kokkos implementations have not been able to reach that level of speed performance. We have since abandoned Kokkos multidimensional Views for the moment, instead using single-dimension Kokkos views in order to have finer control over the memory layout. While this approach is yielding promising results with Kokkos OpenMP, both the Kokkos Cuda and manual Cuda implementations have been disappointingly slow. In some cases this is due to a low ratio of FLOPS to memory reads, while in others we need to work on further coalescing our memory accesses.

We are continuing to work on getting speed-up from Kokkos and manual Cuda, but we feel it would be counterproductive to work on more Intrepid kernels until we have our single prototype kernel producing per-

formance gains on the expected order of magnitude. Therefore, given the amount of time left in the semester, it is likely that we will have to settle for presenting our prototype kernel 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 speed-up. 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 speed-up, 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 speed-up. 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 speed-up. 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:

- Finish a couple functions within Intrepid ArrayTools with expected speed-up to show a Kokkos proof of concept
- Log our experiences with Kokkos on the team wiki
- Finish creating a Kokkos version of Intrepid
- 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.