Computer Science Clinic

Midyear Report for
Sandia National Laboratories

Parallelizing Intrepid with Kokkos

November 23, 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. While we have not been able to get as much speed-up as we wanted so far, 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 and OpenMP for Intrepid tensor contractions. However, our prototype Kokkos implementations have lagged in speed performance, even when taking into account coalescence and memory layout. 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.

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 performance 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 obstacles as previously mentioned. The way that we plan on finishing this job is by focusing our efforts on a couple of the simpler methods in the ArrayTools library in order to get the desired speed-up. Once we have correct Kokkos implementations for a couple of the functions we can apply similar parallel/memory access algorithms to speed-up the remaining functions in Intrepid. Fortunately, the task of our clinic is to show a proof of concept for Kokkos, so it's not critical for us to spend time overhauling the entire Intrepid kernel. If we have a couple functions with great speed-up using Kokkos, then that should be enough to satisfy the purpose of our project.

Throughout this entire process it is important for us to continue to record our thoughts about Kokkos and put these notes on our team wiki so that our liaisons have access to them. This will give Sandia valuable feedback about Kokkos and valuable information on how to improve the user's experience. Logging these notes on the wiki is another of our priorities, since we serve a valuable function 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 by mid February. We do not yet know what our task after completing Intrepid will be, but it will most likely be creating a Kokkos version of a different high performance computing kernel. Depending on how similar the new kernel is to Intrepid in terms of functionality, we plan to tackle the problem as follows. We will begin by finding the simplist functions in the kernel, then create Kokkos versions of those functions, manipulate memory accesses and parallelization points until we achieve reasonable speed-up, then apply the algorithm to the rest of the functions. Our goal is to finish the Intrepid kernel as well as at least one more kernel.

Our stretch goals are to continue to create Kokkos versions of more high performance computing kernels or integrating the Kokkos versions of the kernels back into the original packages. We are leaning towards continuing to create Kokkos versions of more kernels because we will have a better understanding of Kokkos and what is required to gain speed-up. Integrating

the Kokkos versions of the kernels into their original packages brings up new challenges that we have not yet faced and will 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, then repeat with a new high performance computing kernel.