Refactoring EXAALT MD for Emerging Architectures

(the slides are available under "Presentation Materials" in the above URL)
Date: January 15, 2020
Presented by: Aidan Thompson (Sandia National Laboratories), Stan Moore (Sandia National Laboratories), and Rahulkumar Gayatri (NERSC)

Q. When is the recording expected to be online?

A. We should have it online in about a week.

Q. This is my first ECP Webinar, the materials simulation can it simulate adjustments in temperature or just at a set temperature reaction?

A. If temperature is only affecting the velocities of the mass nuclei there are no complicating factors. If shifting the Fermi-Dirac distribution of electronic states, then that undermines the conservative force assumption, which complicates matters. But at temperatures below a few thousand kelvins, this is not a big issue.

Q. How will the approach work for non-homogeneous distribution of atoms in the system?

A. As long as the DFT training data represents the full range of atomic environments e.g. surfaces structures, then the SNAP approach can handle non-homogeneous distributions of atoms just fine.

Q. Is the Kokkos code the reason for this enormous gain for the Haswell or was it the DL Boost in the Xeon CPU (CPU Assumption)? Also, was the GPU code optimized by Mathworks GPU Coder? Snap CPU vs. Snap Titan GPU performance is what I should have worded more clearly in my first question.

A. Improvements in the serial version of the code (Y-array trick) accounted for about 2x improvement, with additional 6x improvement due to improvements in Kokkos version. We have never used the Mathworks GPU Coder.

Q. Eventually, will calculating the distance r_ij (because of the nested for loops) be the most demanding bottleneck of this approach?

A. For simple classical potentials like Lennard-Jones, for sure. In the case of machine learning potentials because of extra calculations after distance calculations, this is not the bottleneck. That's also the case for SNAP. Many nested loops must be executed after the calculation of r_ij.

Notes

- (Slide 2) Today we’re talking about improvements in accuracy of potentials. SNAP simulations in LAMMPS run many small MD simulations. Focus is MD performance on a single node often using GPUs; then scale to more nodes.
- (Slide 3) ”Mira” IBM BG/Q machine is purely CPU machine. Audio recording on ECP website episode 44 describes the improvement.
- (Slide 4) Created stripped-down small TestSNAP to be representative of EXAALT platform. Could then experiment with different GPU strategies, memory footprint, FLOPS.
- (Slide 8) SNAP calculation is a large set of nested sums. Not regular square nested loops, but more like pyramids. Variety of challenges parallelizing nested loops. Difficult to do, but last year made some headway parallelizing nested sums on GPUs.
- (Slide 9) Kokkos supports up to 3 levels of hierarchical parallelism.
- (Slide 11) EXAALT runs replicates and stitches trajectories together.
- (Slide 13) Having a small, proxy TestSNAP was essential to trying lots of ideas without bothering with the baggage that comes with those applications. Has developed into a nice small project for having small, proxy app for other projects; used as a test case. Part of the application suite provided to the BGI complers team; used to test offload implementation of the compiler.
- (Slide 14) Although memory compact implementation, it oversubscribed GPU resources. Counter intuitive solution was increasing memory footprint by creating more kernels.
- (Slide 17) Compacting 5-d arrays into 1-d gave 20x memory footprint improvement.
- (Slide 22) On CPUs row-major matrices preferred to avoid false L1 caching. However we want column major on GPUs so that all threads perform the same calculation during the clock cycle, called “memory coalescing”.
- (Slide 24) Reversed loop-order.
- (Slide 27) Hierarchical parallelism needed row-major layout: after switching to separate kernels then needed to chunk up the loop. Then could add the column-major and loop re-order. All this was only possible after separating the kernels first. Also got rid of shared scratchpad memory.