Interoperability Across GPU Languages and Platforms

Abstract

As researchers look towards exascale computing for large scientific applications, the portability and composability of GPU parallelism becomes a vital endeavor. CUDA and OpenACC are specialized, low-level programming models of GPU hardware, while OpenMP4 and OpenMP4.X are higher-level languages tailored for portability, as details are left to the compiler. How these methodologies perform together is the focus of this research. We assess current interoperability and the performance of a large-scale grid running the Game of Life on GPUs.

Interoperability Results

Interoperability between GPU languages is defined as the ability for each language to affect data on the GPU in an alternating pattern only by the use of its buffer. As shown in Figure 2, grids are allocated on the GPU and the package will apply the rules for 10 generations.

![Interoperability Diagram](image)

Figure 2: Each loop in the figure is what the package will enact on the CPU and GPU

As Nvidia created CUDA and helped develop OpenACC, these two languages are interoperable using a PGI compiler version greater than 12.6. To test this, two packages were written in C, one in CUDA and OpenACC, which apply the rules. Interoperability is also valid for Fortran with CUDA+OpenACC.

![Interoperability Table](image)

Figure 3: This is our current interoperability among the four major GPU languages

Productivity vs. Performance

Figure 4 shows a comparison of development productivity to performance. A fully integrated CUDA version of the test code is 27 times faster than pure OpenACC, but requires 70 lines of code out of which are just to generate pointers. This only applies to the serial case. For the parallel case, there is additional code required to put and pull halo values for MPI.

![Productivity vs. Performance Graph](image)

Figure 4: Lines of serial code versus total runtime for 6.4x10^10 elements

Performance Results

The timings were run on two types of GPUs: Nvidia’s Tesla K40m (on Intel Xeon E5-2660 @ 2.20 GHz processor) and Tesla M2090 (Intel Xeon E5-2670 @ 2.60 GHz).

![Performance Graph](image)

Figure 5: These are the runtime vs. number of elements results for Tesla K40m

Future Directions: Incorporating OpenGl to pass buffers between CUDA and OpenCL. Use Unified Memory Access (UVA) for multiple MPI ranks to interact with the GPU, and among GPUs, for better performance.

Conclusion

With large multi-physics codes, inter-language operability between physics packages, such as ocean and atmospheric models, is necessary to model real-world problems. While the conclusions presented here show that there is a significant benefit of moving to a lower-level language such as CUDA, there is some benefit to including packages where only OpenACC present.

Interoperability Across GPU Languages and Platforms

References

2. Larkin, Jeff. OpenACC Interoperability Tricks, PGI Insider (2013)
4. OpenACC Interoperability Tricks, PGI Insider (2013)

Acknowledgments

The authors thank Jeff Larkin, Software Engineer for NVIDIA, for his insight. The authors also thank LANL Institutional Computing Resources to access to their clusters. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 1321880. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Los Alamos National Laboratory, Los Alamos, NM 87545