GPU PROGRAMMING CONSIDERATIONS FOR APPLIED CFD

JEFF LAYTON*, JOE EATON*, NIKOLAY SAKHARNYKH*, AND STAN POSEY*

* NVIDIA Corporation
2788 San Tomas Expressway, Santa Clara, CA 95051
e-mail: jelayton@nvidia.com, web page: http://www.nvidia.com

Key words: GPU, CFD, CUDA, OpenACC, AmgX, HPGMG.

Summary. This paper examines GPU parallelization strategies for various CFD methods utilizing GPU-based solver libraries, the directives-based programming model OpenACC, and the NVIDIA GPU programming environment CUDA.

1 INTRODUCTION

Current trends in high performance computing (HPC) are advancing towards the use of graphics processing units (GPUs) to achieve speed-ups for numerical operations that are common in solvers for applied computational fluid dynamics (CFD). GPU implementations of CFD software provide additional fine-grain, second-level co-processor parallelism under existing distributed memory, first-level parallelism on CPUs. The programming environment for NVIDIA GPUs offers three strategies of libraries, directives, and a direct programming language, that consider HPC performance, CPU portability, and ease in programming effort.

2 GPU PARALLEL LIBRARIES

Parallel iterative sparse solvers are widely used for simulations that deploy implicit schemes, and have become a standard for commercial CFD software owing to their efficiency in computation and storage, and application to mostly resolve incompressible flow fields. To aid adoption and ease-of-programming for GPUs in applied CFD, complete solver libraries have been developed by NVIDIA that specialize in multi-grid solvers among other methods.

2.1 AmgX Solver Library

The AmgX¹ solver library developed by NVIDIA and first published in 2014, provides algebraic multi-grid schemes for unstructured meshes, and includes parallelization of all phases of AMG including hierarchy construction and ILU factorization and solve. Figure 1. provides an example of AmgX performance for speedups vs. HYPRE on the Florida sparse matrix collection. Results demonstrate as much as a 7x speedup for the AmgX solver on the Tesla K40 GPU vs HYPRE on a 10-core Xeon E5-2690 CPU.

2.2 HPGMG Library

Since 2016, NVIDIA has been developing the HPGMG² solver library for geometric multi-grid methods which are more efficient than AMG for structured grids since they benefit from...
the additional information provided by the geometric representation of the problem. An example of HPGMG and CPU + GPU scalability performance is provided in Figure 2, with results that show efficient scalability to 16000+ nodes on the DOE ORNL system Titan.

![Figure 1: GPU Speedups of AmgX vs. HYPRE for the Florida Sparse Matrix Collection.](image1)

![Figure 2: GPU Scalability of HPGMG on the U.S. DOE Titan System.](image2)

3 OPENACC DIRECTIVES

The opportunity for ‘drop-in’ acceleration of code execution ‘hot spots’ provided by GPU libraries is the first step towards full application acceleration. The next step would involve selection of a programming model depending on the state of the code and HPC objectives. For legacy CFD codes with objectives of performance portability with CPU platforms, the use of OpenACC compiler directives is the most common approach. These directives, which appear as comments to a CPU compiler, can be thoughtfully inserted into the native language of a CFD code (Fortran, C++, etc.) around parallel loops, that instruct the OpenACC compiler to construct parallel operations for those loops on the GPUs. Examples of OpenACC use include FINE™/Turbo from NUMECA and TACOMA from General Electric.

4 NVIDIA CUDA

A programming model that is common with new CFD code development for the GPU is the use of the CUDA programming language from NVIDIA. CUDA is also the back-end for many domain specific language (DSL) developments like Kokkos, that is popular with CFD codes that require performance portability. Examples include RAPTOR from U.S. DOE.

REFERENCES