V European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2010
J. C. F. Pereira and A. Sequeira (Eds)
Lisbon, Portugal,14-17 June 2010

PORTING OF FEFLO TO GPUS

Andrew Corrigan*[1], Fernando Camelli*, Rainald Löhner*, and Fernando Mut*

*Center for Computational Fluid Dynamics,
Department of Computational and Data Sciences,
George Mason University,
Fairfax, VA 22030, USA
e-mail: {acorriga,fcamelli,rlohner,fmut}@gmu.edu

Key words: Graphics hardware, GPUs, Unstructured Grids

Abstract. *This paper describes the porting of a substantial portion of FEFLO to GPUs FEFLO is an adaptive, edge-based finite element code for the solution of compressible and incompressible flow, which is primarily written in Fortran 77 and has previously been ported to vector, shared memory parallel and distributed memory parallel machines. Due to the large scale of FEFLO and the likelihood of human error in porting, a specialized Python script, based on FParser (Peterson, 2009), was written to perform automated translation from the OpenMP-parallelized edge and point loops to GPU kernels implemented in CUDA, along with GPU memory management. The results of verification benchmarks performed and preliminary performance results will be presented.*

1 Introduction

Graphics processing units (GPUs) are becoming a mainstream platform for high performance computational fluid dynamics. The NVIDIA Tesla 10-series GPUs, released in 2008, achieve nearly one teraflop of peak performance, or roughly an order of magnitude higher peak performance than high-end CPUs [12]. Furthermore, the upcoming NVIDIA Tesla 20-series GPUs will feature 512 CUDA cores, more than doubling the number of cores present in the Tesla 10-series, adding a data-parallel cache, and improving peak double-precision performance by a factor of 8x [13].

Due to such a substantial performance advantage of GPUs over CPUs, there has been a great deal of research investigating the implementation of CFD codes on GPUs. Much of this effort has been directed towards the case of structured grid based solvers. These solvers are particularly amenable to GPU implementation due to their regular memory access pattern. Work in this area includes that of Brandvik and Pullan [1–3], who have developed two and three dimensional Euler and Navier-Stokes solvers for GPUs, with support for multiple GPUs via MPI, and achieved an order of magnitude gain in performance.

[1]Currently at the Center for Reactive Flow & Dynamical Systems, Laboratory for Computational Physics and Fluid Dynamics, Naval Research Laboratory, Washington, DC 20375, USA

Göddeke et al. [8] have implemented a multi-level, globally unstructured, locally structured, Navier Stokes solver. LeGresley et al. [11] have implemented a multi-block Euler solver for simulating a hypersonic vehicle configuration, while Cohen and Molemaker [4] have implemented a 3D finite volume Boussinesq code in double precision. Further work on regular grid solvers includes that of Phillips et al. [16], who have developed a 2D compressible Euler solver on a cluster of GPUs, and Jacobsen, Thibault, and Senocak [9, 21], who have implemented a 3D incompressible Navier Stokes solver for GPU clusters.

Additionally, there has been increased interest in running unstructured grid based CFD solvers. Achieving good performance for such solvers is more difficult due to their data-dependent, irregular memory access pattern. Work in this area includes that of Klöckner et al. [10], who have implemented discontinuous Galerkin methods over unstructured grids. In previous work [5] the present authors presented results on running an element-based, finite volume Euler solver on a Tesla 10 series card, which achieved a speedup of nearly a factor of 10x over an OpenMP parallelized CPU code running on a quad-core Intel CPU.

The purpose of the present effort is to obtain such as performance gain in the context of a code used to perform productions runs. The code in consideration is FEFLO, an adaptive, edge-based finite element code for the solution of compressible and incompressible flow. This code has a long history of relevant applications involving compressible flow simulations in the areas of supersonic jet noise, transonic flow, store separation, and blast-structure interaction, as well as incompressible flows including free-surface hydrodynamics, dispersion, and patient-based haemodynamics. The code is written primarily in Fortran 77, and has already been ported to vector, shared memory and distributed memory machines using OpenMP for the former and MPI for the latter.

In previous work [6] the present authors provided a preliminary report of progress made implementing a Python script to automatically translate FEFLO to GPUs, including (i) converting OpenMP loops to CUDA kernels while exposing finer-grained parallelism, (ii) detecting GPU arrays and enforcing consistency across subroutine calls, (iii) using an array layout appropriate for meeting coalescing requirements, (iv) overriding certain subroutines with custom implementations, (v) automatically handling GPU array IO and memory transfer, and (vi) tracking subarrays across subroutine calls. This Python script now produces code which runs from start to finish on the GPU. In this presentation, we will review the rationale for this approach, along with its key features and limitations, describe its application to FEFLO, and present the results of a variety of runs performed for validation purposes, along with preliminary performance measurements.

REFERENCES

[1] T. Brandvik and G. Pullan - Acceleration of a Two-Dimensional Euler Flow Solver Using Commodity Graphics Hardware; *J. Proc. of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, Vol. 221, 2007, pp. 1745–1748.

[2] T. Brandvik and G. Pullan - Acceleration of a 3D Euler Solver Using Commodity Graphics Hardware; 46th AIAA Aerospace Sciences Meeting and Exhibit, January 2008.

[3] T. Brandvik and G. Pullan - An Accelerated 3D Navier-Stokes Solver for Flows in Turbomachines; Proceedings of GT2009 ASME Turbo Expo 2009: Power for Land, Sea and Air June 8-12, 2009.

[4] J.M. Cohen, M.J. Molemaker - A Fast Double Precision CFD Code using CUDA; *Parallel CFD*, (2009).

[5] A. Corrigan, F. Camelli, R. Löhner, and J. Wallin, "Running Unstructured Grid Based CFD Solvers on Modern Graphics Hardware," *19th AIAA Computational Fluid Dynamics Conference*, No. AIAA-2009-4001, June 2009.

[6] A. Corrigan, F. Camelli, R. Löhner, and F. Mut, "Porting of an Edge-Based CFD Solver to GPUs," *48th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, No. AIAA 2010-522, January 2010.

[7] E. Cuthill and J. McKee - Reducing the Bandwidth of Sparse Symmetric Matrices; *Proc. ACM Nat. Conf.*, New York 1969, 157-172 (1969).

[8] D. Göddeke, S.H.M. Buijssen, H. Wobker, and S. Turek - GPU Acceleration of an Unmodified Parallel Finite Element Navier-Stokes Solver; *High Performance Computing & Simulation*, 12-21, (2009).

[9] Jacobsen, D., Thibault, J., and Senocak, I., "An MPI-CUDA Implementation for Massively Parallel Incompressible Flow Computations on Multi-GPU Clusters," *48th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, No. AIAA 2010-522, January 2010.

[10] Klöckner, A., Warburton, T., Bridge, J., and Hesthaven, J. S., "Nodal Discontinuous Galerkin Methods on Graphics Processors", *Journal Computational Physics*, Vol. 228, 2009, pp. 7863–7882.

[11] P. LeGresley, E. Elsen, E. Darve - Large calculation of the flow over a hypersonic vehicle using a GPU; *J. Comput. Phys.* 227, 10148–10161, (2008).

[12] NVIDIA Corporation, *NVIDIA CUDA Compute Unified Device Architecture 3.0 Programming Guide*, 2010.

[13] NVIDIA Corporation, *Fermi Compute Architecture White Paper*, 2009.

[14] Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Krger, J., Lefohn, A. E., and Purcell, T. J. - A Survey of General-Purpose Computation on Graphics Hardware; *Computer Graphics Forum*, Vol. 26, No. 1, 2007, pp. 80–113.

[15] Peterson, P. - F2PY: a tool for connecting Fortran and Python programs; *Int. J. Computational Science and Engineering*, Vol. 4, No. 4, 296-305 (2009).

[16] Phillips, E., Zhang, Y., Davis, R., and Owens, J., "Rapid Aerodynamic Performance Prediction on a Cluster of Graphics Processing Units," *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, No. AIAA 2009-565, January 2009.

[17] The Portland Group, *PGI Fortran & C Accelerator Programming Model*, 2009.

[18] C. Scheidegger, J. Comba, R. C. - Practical CFD simulations on the GPU using SMAC; *Computer Graphics Forum*, Vol. 24, 2005, pp. 715–728.

[19] D. Sharov, H. Luo, J.D. Baum and R. Löhner - Implementation of Unstructured Grid GMRES+LU-SGS Method on Shared-Memory, Cache-Based Parallel Computers; *AIAA*-00-0927 (2000).

[20] K.M. Shyue - An Efficient Shock-Capturing Algorithm for Compressible Multicomponent Problems; *J. Comp. Phys.* 142, 208242 (1998).

[21] Thibault, J., and Senocak, I., "CUDA Implementation of a Navier-Stokes Solver on Multi-GPU Desktop Platforms for Incompressible Flows," *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, No. AIAA 2009-758, January 2009.