DEVELOPING A FUTURE-PROOF CFD CODE

D. Curran*, S. McIntosh-Smith*, C.B. Allen†, and D. Beckingsale††

* Department of Computer Science, † Department of Aerospace Engineering,
University of Bristol, Bristol, BS8 1TR, England, UK.
e-mail: dc8147@bristol.ac.uk, c.b.allen@bristol.ac.uk, simonm@cs.bris.ac.uk.
†† Department of Computer Science, University of Warwick, UK.
e-mail: dab@dcs.warwick.ac.uk.

Key words: Parallel CFD, GPU porting, Source-to-source translation, Parallel scaling

Abstract. The future-proof development of a compressible finite-volume RANS code is considered. An existing domain-decomposed Fortran 90/MPI production application has been redeveloped in C including a generic data communication harness, to allow exploitation of diverse hardware platforms, and future-proof the code for new highly-parallel hardware and languages. This includes development of a kernel stencil DSL and source-to-source translation for the auto generation of OpenMP, MPI, OpenCL, and CUDA code versions, and heterogeneous combinations thereof.

1 INTRODUCTION

The standard approach to parallel CFD, until fairly recently, has been domain decomposed simulation via distributed memory, using MPI, or hybrid OpenMP/MPI implementations on larger memory machines, most commonly exploiting the x86 architecture. However, hardware progression, particularly in terms of accelerator cards, means processing per cost unit and power unit can be significantly improved over conventional CPU-only approaches, so there is huge interest in adopting these technologies. Legacy code and uncertainty over available languages has lead to some inertia in adoption, but there has been a recent increase in porting CFD tools to accelerators, and most fidelities of mainstream CFD codes have now been adapted to exploit accelerators; for example RANS [1], LES [2], and high-order spectral methods [3].

The code considered here is an explicit finite-volume multigrid Euler and RANS code, and the work presented is based on a complete rewrite of the original Fortran90 source, from data structure upwards, in an attempt to optimise performance of the code for multiple hardware platforms, enabling heterogeneous exploitation of mixed hardware, and future-proofing the code for future generations of technology and languages.

2 DEVELOPMENT STRATEGY

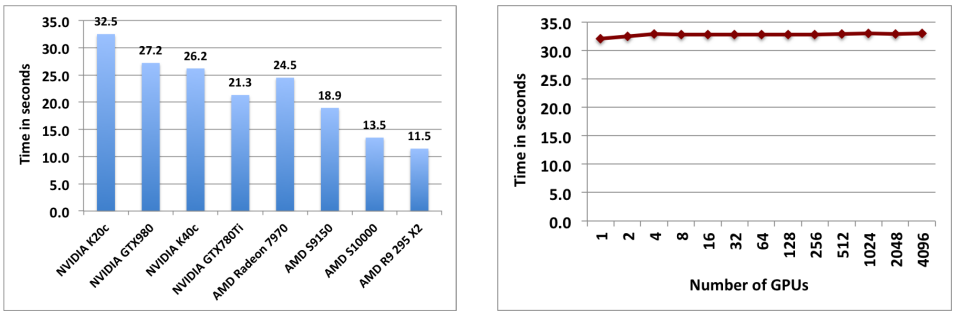
The original code is written in Fortran90/MPI and has been shown to scale linearly to over 1000 x86 cores [4]. The first objective of the redevelopment was to ensure any future parallel hardware could be exploited. There are no clear winners in which parallel programming paradigm can be used to target all of the interesting hardware; Intel favours OpenMP/MPI, Nvidia favours CUDA and OpenACC, while AMD favours

OpenCL. Hence, while all three do support OpenCL on their CPUs and GPUs, it was essential the approach here would make it easy to adapt to any new parallel programming paradigms that might emerge in the future.

To address these goals an approach was adopted where the application uses a Domain Specific Language (DSL) to express the main mathematics behind the algorithm, coupled with a parallel code generator which emits optimised, efficient code for a very diverse range of parallel computer architectures and languages. This approach shares some similarity with recent work on the OPS parallel library [5].

3 PERFORMANCE

The code has been tested for performance across a range of hardware, and for scaling performance on upto 4096 GPUs on the Oak Ridge Titan machine. The problem size was scaled to match the number of GPUs; the largest case was over 3.4 billion mesh points. Figure 1 (a) shows the run time of the case on various GPUs, and (b) shows the scaling with GPUs. A single threaded run on a Haswell 3.8GHz core required 620 seconds.



(a) Performance on various GPUs. (b) Weak scaling on Titan.

Figure 1: Performance of ROTORSIM.

4 ONGOING WORK

The final paper will include: full background and literature review; more details of development strategy; performance comparison across many-core processors from Nvidia, AMD and Intel; more weak and strong scaling results on large accelerated systems, including Piz Daint at CSCS and Titan at Oak Ridge.

REFERENCES

[1] Thomas, S., Baeder, J., AIAA paper 2013-2853, San Diego, 2013.

[2] Patnaik, G., Corrigan, A., Obenschain, K., Schwer, D., Fyfe, D., AIAA paper 2012-563, Florida, 2012.

[3] Zimmerman, B.J., Wang, Z.J., Visbal, M.R. , AIAA paper 2013-2941, San Diego, 2013.

[4] Allen, C.B., Int J Numerical Methods in Engineering. Vol. 68(6), 2006.

[5] István Z. Reguly, Gihan R. Mudalige, Michael B. Giles, Dan Curran, and Simon McIntosh-Smith. Proceedings WOLFHPC '14. IEEE Press, Piscataway, NJ, USA, 2014.

[6] S.N. McIntosh-Smith, M. Boulton, D. Curran and J.R. Price, ISC, Leipzig, June 2014.