Rolls Royce Hydra CFD Code on GPUs using OP2 Abstraction

I. Z. Reguly, G. R. Mudalige, C. Bertolli, M. B. Giles, A. Betts, P. H. J. Kelly, D. Radford

***Abstract*—Hydra is an industrial CFD application used for the design of turbomachinery, now automatically accelerated by GPUs through the OP2 domain specific "active library" for unstructured grid algorithms. From the high-level definition, either CPU or GPU code is generated, applying optimisations such as conversion to Structure-of-Arrays, use of the read-only cache or the tuning of block sizes automatically. A single GPU is over 2 times faster than the original on a server-class CPU, we demonstrate excellent strong and weak scaling, evaluated up to 4096 CPU cores or 16 GPUs.**

Hydra is a full-scale industrial CFD application used for the design of turbomachinery at Rolls Royce plc [1], [2]. It consists of over 300 parallel loops with a code base exceeding 50K lines and is capable of performing complex simulations over highly detailed unstructured mesh geometries, solving the Reynolds-Averaged Navier-Stokes (RANS) equations, often taking on the order of weeks to complete highly complex simulations. Unlike simpler structured-mesh applications, which feature high speed-ups when accelerated by modern processor architectures, such as multi-core and many-core processor systems, Hydra presents major challenges in data organization and movement that need to be overcome for continued high performance on emerging platforms.

We present research in achieving this goal through the OP2 [3], [4] domain-specific high-level framework. OP2 targets the domain of unstructured mesh problems and follows the design of an active library using source-to-source translation and compilation to generate multiple parallel implementations from a single high-level application source for execution on a range of back-end hardware platforms. To our knowledge this research presents the first application of such a high-level framework to a full scale production code.

Specifically we show that (1) different parallel implementations can be achieved with an active library framework, even for a highly complicated industrial application such as Hydra, and (2) that different optimizations targeting GPU architectures can be applied to the whole application, seamlessly, reducing developer effort and increasing code longevity.

We discuss a number of optimizations to the CUDA backend and the execution scheme, including tuning the thread block size, using the texture cache and and applying a data transformation from Array-of-Structs (AoS) to Struct-of-Arrays, these can be applied without user intervention. We show the effects of these optimizations, demonstrating a 2x performance improvement using a single NVIDIA K20c GPU over the original code running on a dual-socket Xeon processor.

We perform strong scaling tests of the CPU backend on HECToR (Cray XE6), on up to 4096 cores, and of the GPU backend on a smaller GPU cluster (JADE), on up to 16 cards. Comparing a HECToR node (32 cores) with a JADE node (2 K20m GPUs) we see up to 2x improvement by using GPUs [5].

Finally, we introduce a hybrid CPU-GPU execution scheme that can utilize all hardware resources in a system. We show that the most important challenge is load balancing between the CPU and the GPU, due to the differences in relative performance of different computational loops. A 15% speedup over a single GPU is demonstrated when utilizing the CPUs in the system as well.

Our results provide evidence of how high-level frameworks such as OP2 enable portability across a wide range of contrasting platforms and their significant utility in achieving near-optimal performance without the intervention of the application programmer.

References

[1] P. I. Crumpton and M. B. Giles, "Multigrid Aircraft Computations Using the OPlus Parallel Library," *Parallel Computational Fluid Dynamics: Implementations and Results Using Parallel Computers*, vol. -, pp. 339–346, A. Ecer, J. Periaux, N. Satofuka, and S. Taylor, (eds.), North-Holland, 1996.

[2] M. B. Giles, M. C. Duta, J. D. Muller, and N. A. Pierce, "Algorithm Developments for Discrete Adjoint Methods," *AIAA Journal*, vol. 42, no. 2, pp. 198–205, 2003.

[3] M. B. Giles, G. R. Mudalige, Z. Sharif, G. Markall, and P. H. J. Kelly, "Performance analysis and optimization of the OP2 framework on many-core architectures," *The Computer Journal*, vol. 55, no. 2, pp. 168–180, 2012.

[4] M. B. Giles, G. R. Mudalige, B. Spencer, C. Bertolli, and I. Reguly, "Designing OP2 for GPU Architectures," *Journal of Parallel and Distributed Computing*, vol. 73, pp. 1451–1460, November 2013.

[5] I. Z. Reguly, G. R. Mudalige, C. Bertolli, M. B. Giles, A. Betts, P. H. J. Kelly, and D. Radford, "Acceleration of a Full-scale Industrial CFD Application with OP2," *submitted to ACM Transactions on Parallel Computing*, 2013, available at: http://arxiv.org/abs/1403.7209.