Rolls-Royce Hydra CFD Code on GPUs using OP2 Abstraction

I. Z. Reguly¹, G. R. Mudalige¹, C. Bertolli¹, M. B. Giles¹, A. Betts¹, P. H. J. Kelly¹, D. Radford¹
¹Oxford e-Research Centre, University of Oxford, UK
²IBM TJ Watson Research Center, USA
³Department of Computing, Imperial College London
⁴Rolls-Royce plc.

Rolls-Royce Hydra

A highly complex and configurable full-scale industrial application used for the simulation and design of turbomachinery components. Development started over 15 years ago in Fortran, relying on the OPlus library to run on distributed memory clusters, it consists of 300+ parallel loops. Guiding equations are the Reynolds-Averaged Navier-Stokes second-order PDEs by default using a 5-step Runge-Kutta method for time-marching, accelerated by multigrid and block-Jacobi pre-conditioning.

OP2 Abstraction for Unstructured Grid Computations

Sets:
- op_decl_set(num_cells, "cells")
- op_decl_set(num_edges, "edges")
- op_decl_set(num_vertices, "vertices")

Mappings between sets:
- Cells2Edges: (Cells, Edges)
- Edges2Vertices: (Edges, Vertices)

Data on sets:
- Coordinates on vertices:
  - v_1, v_2, v_3
- Flow variables on cells:
  - u, c, d, e

Based on this information, OP2 can automatically handle data dependencies and generate code for parallel execution on a range of heterogeneous architectures

Platform-specific code generation and compilation

OP2 Source-to-Compiler (C/C++ or FORTRAN API)

OP2 Platform Specific Optimized Application Files

Conventional Compiler + compiler flags
(e.g. -Ic, -lv, -lgcc)

Platform Specific Binary Executable

Hardware

Unstructured Mesh Application using OP2 (C/C++ or FORTRAN API)

OP2 Platform Specific Optimized Back-end libraries

With OP2, it is possible to utilize the CPU and the GPU at the same time. The key challenge is to find the right load balance when there are performance differences between loops on different hardware: with a good balance, 15% speedup can be achieved over a single GPU by utilizing the CPUs in the system as well.

Timings from a machine with a dual-sOCKET Intel Xeon E2640 CPU and 2 NVIDIA Tesla K20c GPUs.

Timings from a machine with a dual-socket Intel Xeon E2640 CPU and 2 NVIDIA Tesla K20c GPUs.

Optimizations can be enabled in the code generator and parameters can be automatically tuned:
- Transposing data to Structure of Arrays (SoA)
- Moving read-only data through the texture cache
- Tuning thread block size and register count

SoA is especially advantageous with high dimensional datasets - gives coalesced memory accesses and decreases cache contention. Using the texture cache helps maximize the bandwidth and gives data reuse for indirect accesses.

Fully automated distributed execution:
- Partitioning using ParMetis or PT-Scotch
- Latency hiding
- Halo exchanges and redundant execution to facilitate owner-compute approach

Weak scaling performance

With good utilization up to 2 times performance gain over fully utilized HECToR nodes (32 cores), at low node counts when strong scaling, and maintained weak scaling doubling problem size when doubling node count.

Strong scaling performance

Timings for the CPU from HECToR (32 AMD Opteron cores per node) and for the GPU from Jade (2 Tesla K20m cards per node), strong scaling with 800k vertex mesh, weak scaling with 500k vertex per node

Parallelization happens on three levels: (1) distributed memory, using MPI and the standard owner-compute approach with halo exchanges, (2) coarse-grained shared memory, for CUDA thread blocks where a partition is broken up into blocks which are colored based on potential data races and then executed by color; and (3) fine-grained shared memory, for CUDA threads in the same thread block, where set elements are colored based on potential data races, accesses are then serialized by color. Blocks are classified as interior and boundary, based on whether they need data from a neighboring MPI process, latency hiding is facilitated by executing interior blocks and communications at the same time.

Acknowledgements
This research has been funded by the UK Technology Strategy Board and Rolls-Royce plc. through the IsoMet project, the UK Engineering and Physical Sciences Research Council projects EP/H000705/1, EP/K000575/1 on “Multi-layered Abstractions for PDEs” and the “Algorithmic, Software for Emerging Architectures” (AS2EnA) EP/I010557/1 project. The authors would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out this work.

Organizing parallelism

MPI boundary
- Owner-compute
- Halo exchanges

Hardware

Hardware

Hybrid CPU-GPU execution

With good utilization up to 2 times performance gain over fully utilized HECToR nodes (32 cores), at low node counts when strong scaling, and maintained weak scaling doubling problem size when doubling node count.

Timings for the CPU from HECToR (32 AMD Opteron cores per node) and for the GPU from Jade (2 Tesla K20m cards per node), strong scaling with 800k vertex mesh, weak scaling with 500k vertex per node

Acknowledgements
This research has been funded by the UK Technology Strategy Board and Rolls-Royce plc. through the IsoMet project, the UK Engineering and Physical Sciences Research Council projects EP/H000705/1, EP/K000575/1 on “Multi-layered Abstractions for PDEs” and the “Algorithmic, Software for Emerging Architectures” (AS2EnA) EP/I010557/1 project. The authors would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out this work.

Single node performance

With good utilization up to 2 times performance gain over fully utilized HECToR nodes (32 cores), at low node counts when strong scaling, and maintained weak scaling doubling problem size when doubling node count.

Timings for the CPU from HECToR (32 AMD Opteron cores per node) and for the GPU from Jade (2 Tesla K20m cards per node), strong scaling with 800k vertex mesh, weak scaling with 500k vertex per node

Hybrid CPU-GPU execution

With good utilization up to 2 times performance gain over fully utilized HECToR nodes (32 cores), at low node counts when strong scaling, and maintained weak scaling doubling problem size when doubling node count.

Timings for the CPU from HECToR (32 AMD Opteron cores per node) and for the GPU from Jade (2 Tesla K20m cards per node), strong scaling with 800k vertex mesh, weak scaling with 500k vertex per node