Abstract
Developers of scientific codes face an important dilemma in order to achieve high performance, it is increasingly necessary to implement low-level optimisations that are specific to a certain hardware. At the same time, there is considerable uncertainty about what programming approaches and hardware are best suited to different problems and how they will change in the future. It would be unfeasible to refactor large codes to every new generation of hardware. Domain-specific languages address this problem by providing a high-level abstraction for a specific class of applications. OP2 defines such an abstraction for unstructured grid computations, that hides the details of parallelism and data movement, and enables efficient mapping of execution to various programming abstractions and hardware.

Rolls-Royce Hydra

- Complex and configurable full-scale industrial application used for the simulation and design of turbomachinery components.
- Equations solved are the Reynolds-Averaged Navier-Stokes second-order PDEs, using a 5-step Runge-Kutta method for time-marching, accelerated by multigrid and block-Jacobi preconditioning.
- Fortran code, uses the OPlus library, consists of 300+ parallel loops.
- OPlus library only supports distributed memory parallelism over MPI, but the latest hardware require the utilization of shared memory programming models (OpenMP, CUDA).
- Adopted Hydra to use OP2 and support modern heterogeneous architectures.

OP2 Abstraction for Unstructured Grid Computations

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

Mappings between sets:
- op_map_set (cells, edges, 4, "edge"
- op_map_set (vertices, "vertex")

Data on sets:
- op_map (vertices, "edge")
- op_map (cells, "partial cells")

Parallel loops:
- Iterate over all the elements in a set in parallel, executing a user-defined "kernel function" on each, passing a number of data arguments, either on the iteration set or through at most one level of indirection, describing the type of access (read / write / increment).

Conclusion
With good utilization up to 2 times performance gain over fully utilized HECToR nodes (32 cores), at low node count when strong scaling, and 85% when weak scaling - doubling when doubling node count.

Timings from a machine with a dual-sOCKET Intel Xeon E2660 CPU and 2 NVIDIA Tesla K20c GPUs.

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.

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.

OP2 provides code maintainability and future-proofing to the application developers.