Geophysical Research Abstracts
Vol. 16, EGU2014-16150, 2014
EGU General Assembly 2014

Implementing hybrid MPI/OpenMP parallelism in Fluidity

Gerard Gorman (1), Michael Lange (1), Alexandros Avdis (1), Xiaohu Guo (2), Lawrence Mitchell (3), and Michele Weiland (4)

(1) Dept Earth Science and Engineering, Imperial College London, London SW7 2AZ, UK, (2) Scientific Computing Department, Science and Technology Facilities Council, Daresbury Laboratory, Warrington WA4 4AD, UK, (3) Computing, Imperial College London, London SW7 2AZ, UK, (4) EPCC, The University of Edinburgh, Edinburgh EH9 3JZ, UK

Parallelising finite element codes using domain decomposition methods and MPI has nearly become routine at the application code level. This has been helped in no small part by the development of an eco-system of open source libraries to provide key functionality, for example SCOTCH for graph partitioning or PETSc for sparse iterative solvers. As we move to an era where pure MPI no longer suffices, application developers cannot only focus on the application code, but must consider the full software stack. In the case of Fluidity (an open source control volume/finite element general purpose fluid dynamics code) the decision to improve parallel efficiency by moving to a hybrid MPI/OpenMP programming model it became necessary to get involved in extending 3rd party open source libraries, specifically PETSc, in addition to the application code itself. The effort involved in re-engineering a large application code highlights the fact that as computing platforms continue their advance towards low power many core processors, the software stack must also develop at a similar pace or application codes will suffer.
In this presentation we will illustrate the steps required to re-engineer Fluidity to achieve good parallel efficiency when using MPI/OpenMP. We identify performance pitfalls when using Fortran features such as automatic arrays in a multi-threaded context, as well as poor data locality on NUMA platforms. A significant proportion of the computational cost is in the sparse iterative solvers. For this we collaborated with the development team at Argonne National Laboratory to add OpenMP support to PETSc. We will present performance results for both the application as a whole, as well as for key individual components such as matrix assembly and the solvers. We also show that while we did not explicitly target I/O for optimisation here, its performance is nonetheless greatly improved because of fewer processes accessing the file system. One of the main remaining challenges for the community at large is the development of efficient preconditioners that will scale up to high core counts when using a hybrid parallel programming model.