Computational Rheological Modelling

Introduction

The HPC Wales project HPCW035 – "HPC for Computational Rheological Modelling" – is built around the use of a Rheology code produced by the Swansea group of Prof. M.F. Webster in the School of Engineering. This code solves 2D and 3D simulations of non-Newtonian fluids on unstructured meshes. Example results are shown in the figure, right[1].

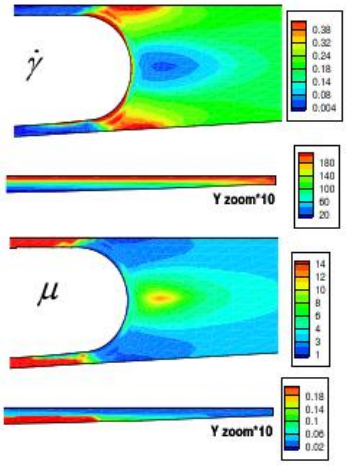

Figure 1 : Shear rates, top, and viscosity in coating flows

Assessment work was undertaken to look at both the serial performance and the parallel scalability of the code. The purpose of this work was part of the HPC Wales effort to increase the overall efficiency of software running on the system in order to maximise useful computational science from CPU hours used.

The parallelism in the code was initially performed using a legacy PVM[2] implementation. HPC Wales funding was used to help modernise the code to using MPI. A significant part of the work was therefore assessing the newly implemented parallel algorithms undertaken by Prof. Webster's team and making suggestions on how these may be enhanced further.

PVM and MPI

The 1980s saw a proliferation of parallel programming environments for the different manufacturers producing hardware, with each vendor having their own library. This meant that codes were never portable from one system to another. Oak Ridge National Laboratory attempted to remedy this in 1989 by developing a message passing library called *Parallel Virtual Machine* (PVM), released publically in 1993. Between 1992 and 1994 manufacturers, researchers and industry all collaborated on developing a standard library for distributed memory computing, known as the *Message Passing Interface* (MPI). Since then MPI has been universally adopted and supported across all hardware vendors, with the standard still growing, currently at MPI-3.

[1] Reverse roll-coating flow: A computational investigation towards high-speed defect free coating, Belblidia, F. Tamaddon-Jahromi, H.R. Echendu, S.O.S. Webster, M.F., *Mechanics of Time-Dependent Materials.* (2013)
[2] Parallel Virtual Machine, http://www.csm.ornl.gov/pvm/

Ewrop & Chymru: Buddsoddi yn eich dyfodol
Cronfa Datblygu Rhanbarthol Ewrop
Europe & Wales: Investing in your future
European Regional Development Fund
ERDF
Llywodraeth Cymru
Welsh Government

The differences between the two systems are not huge, with PVM having implemented several ideas, such as one-sided communication, before MPI. However MPI always had a greater breadth of functions available for users, and processes were always independent rather than seen as part of a virtual machine. As such MPI had lower communication overheads and hence better performance on massively parallel computers, with no need for a master process or daemons running on hosts. Today PVM has almost no supported tools for profiling or optimisation, nor is it officially supported by any major vendor and the open source version has not been updated in years. As such the remaining PVM codes around will benefit greatly from a move to optimised MPI libraries, allowing them to take advantage of high speed interconnects.

Serial performance

In order to analyse fully the serial performance of the rheology code routine and line level profiling was undertaken. In addition the call graphs to the most computationally demanding routines were also examined. A small number of very high cost routines were identified and changes suggested that helped increase vectorization. An example detailed how a 10% saving in overall runtime could be obtained through pre-computation of variables outside loops.

Parallel scalability

The parallel scalability work looked at both the PVM and MPI codes. The initial MPI version was found to not scale as well as the original PVM version. Detailed results, including the use of the profiling tool Scalasca[3], were presented, together with an analysis of the issues that would impact on parallel performance, plus pointers to the simple rewriting necessary to avoid these bottlenecks. A plan for implementing these suggested changes was also presented to the code authors.

The biggest issue concerning the extension to large numbers of cores was found to be the use of the master-slave model. Again profiling data, this time using the MPE library[4] and the Jumpshot[5] trace visualization tool, was used to highlight the limits of this approach. A more extensive rewriting of the software to avoid using this model was suggested should the authors plan to extend to large computational domains necessitating greater core counts. Other advice given included software engineering best practice and advocation of utilisation of existing parallel libraries.

[3] http://www.scalasca.org/
[4] http://www.mcs.anl.gov/research/projects/perfvis/software/viewers/
[5] http://www.mcs.anl.gov/research/projects/perfvis/software/viewers/