Software engineering tools for OP2 project

Mike Giles

September 14, 2010

Abstract

As we begin phase 2 of the OP2 project, with an increased amount of collaboration between different developers, I think it is a good time to consider the software engineering tools we will use.

As an Associate Director of the Oxford e-Research Centre, I'm also interested in this topic to identify and promote best practice in software projects within OeRC.

This is an area in which I am aware that my own knowledge and practice is deficient, so many of my suggestions are based on the recommendations of others, and I will be interested to hear people's opinions on them.

Tools

- Version control

  I've been told that git and the web-based GitHub are much better than the older CVS and SVN.

- Make system

  Currently, we use GNU makefiles, but in the future we should consider using CMake which is platform independent and has CUDA support.

- Compilers

  Currently we are using NVIDIA's `nvcc` for the CUDA code, and `g++` for the C/C++ code, but in the future we will be using PGI's CUDA FORTRAN compiler, and also ought to consider Intel's compilers for multithreaded C/C++/FORTRAN code and to produce AVX executables.

- MPI

  There are several different MPI implementations. We will probably continue to use the one recommended / supplied by the hardware supplier.

  For parallel file I/O, we plan to use Parallel HDF5 which is built on MPI-IO, rather than directly using MPI-IO.

- Parallel debugging

  I think Allinea's DDT is the best choice for both MPI and CUDA applications. At present the CUDA support is only for CUDA version 3.0, but 3.1/3.2 should be supported in the near future

- Performance monitoring and optimisation

  I think Allinea's OPT may be good for MPI applications, but I don't know if it has any support for CUDA.

  Stephen Jarvis' group models the performance of parallel applications using network data obtained from Scalasca and low-level CPU performance data from PAPI.

  Another possibility for low-level optimisation of CPU code (particularly multithreaded code?) is Intel's vtune software.

- Code documentation

  At present, I am happy to write documentation in LaTeX, and that allows me to write different sets of documentation for users and developers.

  However, some people recommend the use of Doxygen and this could be considered to generate the definitive developer reference.

- Build-and-test

  I'm told that Buildbot is the standard choice for automated compilation and testing.

- Coverage testing

  This is something I've never used before, but tools such as gcov and lcov check which bits of your code are being exercised by the testcases used to validate the code.

- Code generation

  Currently, the OP2 code generator is written in MATLAB. This is not an ideal choice; it's simply a language I am comfortable with. Paul Kelly's group may re-write it entirely in rose or it could be ported to python.