Using XML to Specify a Trace Format for MPI Programs

Simon Huband and Chris McDonald
Department of Computer Science & Software Engineering
The University of Western Australia
Crawley, Western Australia 6009, Australia
{huey, chris}@cs.uwa.edu.au

Trace files have long been used to assist correctness debugging and performance debugging of parallel programs. With the advent of implementations of the Message Passing Interface (MPI) standard, parallel and distributed computing has become more common, and thus the need for quality debugging tools has increased. It is important that trace file formats be extensible, flexible, and architecturally independent, the latter particularly if analysis is performed on a different platform to that which generated the trace. In this paper we propose a set of requirements for MPI-based trace libraries, and present a preliminary trace library, tracempi. An important contribution is that this trace format uses the Extensible Markup Language (XML), and XML Schema. By doing so, it is architecturally independent, well defined, and easily extended.

Keywords: MPI, parallel program debugging, XML, trace files
CR Categories: D.1.3 [Concurrent Programming], D.2.5 [Testing and Debugging]

1 INTRODUCTION
By drawing upon multiple processors concurrently, parallel and distributed processing allows us to address computationally intensive tasks for which the use of a single processor is considered inadequate. Some application domains, such as weather forecasting and climate research, demand the power afforded by parallel and distributed processing. Finer grained tasks, such as parallel matrix manipulations and parallel sorting, serve as an important basis for more complicated parallel programs. Across many levels of computing, parallel and distributed processing is an important asset.

With the advent of popular message passing environments such as the Parallel Virtual Machine (PVM) (Geist, Beguelin, Dongarra, Jiang, Manchek, and Sunderam, 1994), the Message Passing Interface (MPI) standard (Message Passing Interface Forum, 1995), and later MPI-2 (Message Passing Interface Forum, 1997), parallel programming has become increasingly standardised and portable. Implementations of MPI exist for architectures ranging from dedicated supercomputers through loosely connected networks of heterogeneous workstations. Parallel programming has become increasingly accessible and thereby increasingly popular.

However, designing and debugging parallel programs remains difficult. Poorly designed parallel programs perform sub-optimally and, compared to sequential programs, are more difficult to code,