

## Scientific Application Development using Eclipse and the Parallel Tools Platform

Greg Watson, Craig E. Rasmussen, CCS-1

**S**cientific application developers face many problems in today's parallel computing environments. Despite the perception that there are few tools supporting high-performance computing, the opposite is in fact true. Unfortunately, the tools are often difficult to use, support only a few platforms, and in many cases, are incompatible with each other.

An integrated development environment (IDE) is generally considered best practice in the majority of the (nonscientific) software development industry. Eclipse is an open-source IDE that provides a portable, robust, commercial quality environment for a wide range of software development activities. It supports multiple languages, including Java, C, C++, and Fortran, provides a syntax-aware editor, code refactoring, incremental code compilation, source-level debugging, and integrated support for source control systems such as CVS and Subversion.

To assist scientific application developers to improve their productivity, we created the Eclipse Parallel Tools Platform (PTP) project (<http://eclipse.org/ptp>). The aim of this project is to establish a common and portable IDE across a wide range of parallel computing platforms. An important aspect of PTP is that it remains agnostic to the tools actually deployed on the machines, such as compilers, linkers, job schedulers, runtime systems, and performance analysis tools. This ensures portability of the platform, and also provides a tool integration framework that allows the individual tools to share data and functionality.

In addition to the standard features supported by the Eclipse platform, PTP adds a range of functionality that enhances the ability of software engineers to develop codes for parallel machines. These features include:

### **Tools to aid MPI and OpenMP programmers.**

PTP includes tools that simplify the development of MPI and OpenMP programs by providing special content assistance and context-sensitive help. In addition, the tools provide static analysis features that provide advanced error checking and analysis of parallel programming constructs.

### **The ability to monitor and control parallel jobs.**

PTP adds support for monitoring the status of a parallel computer system (such as which nodes are available, etc.) and launch parallel jobs onto the system. Figure 1 shows the Eclipse view of a parallel machine. The status and output from processes of the job can be monitored, and the job can be terminated if necessary. Recent work has added support for submitting jobs through job-scheduling systems such as LSF and MOAB.

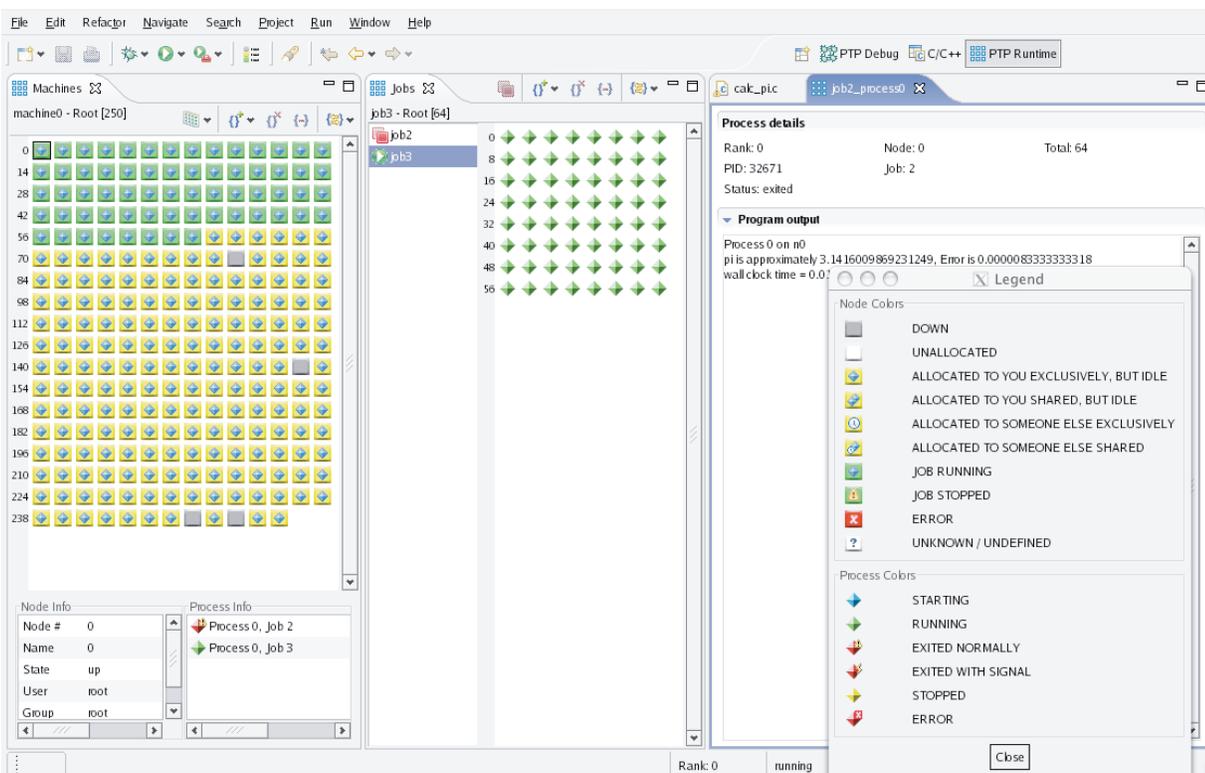
### **The ability to debug parallel programs.**

PTP adds an integrated, scalable, parallel debugger to the IDE. The debugger is launched with the click of a button, and provides the ability to control many processes simultaneously. Figure 2 shows a typical debug session. It supports traditional debugging commands, such as breakpoints, viewing variables, etc., as well as some innovative functions for dealing with multiple processes. The debugger is designed to scale to jobs with many thousands of processes.

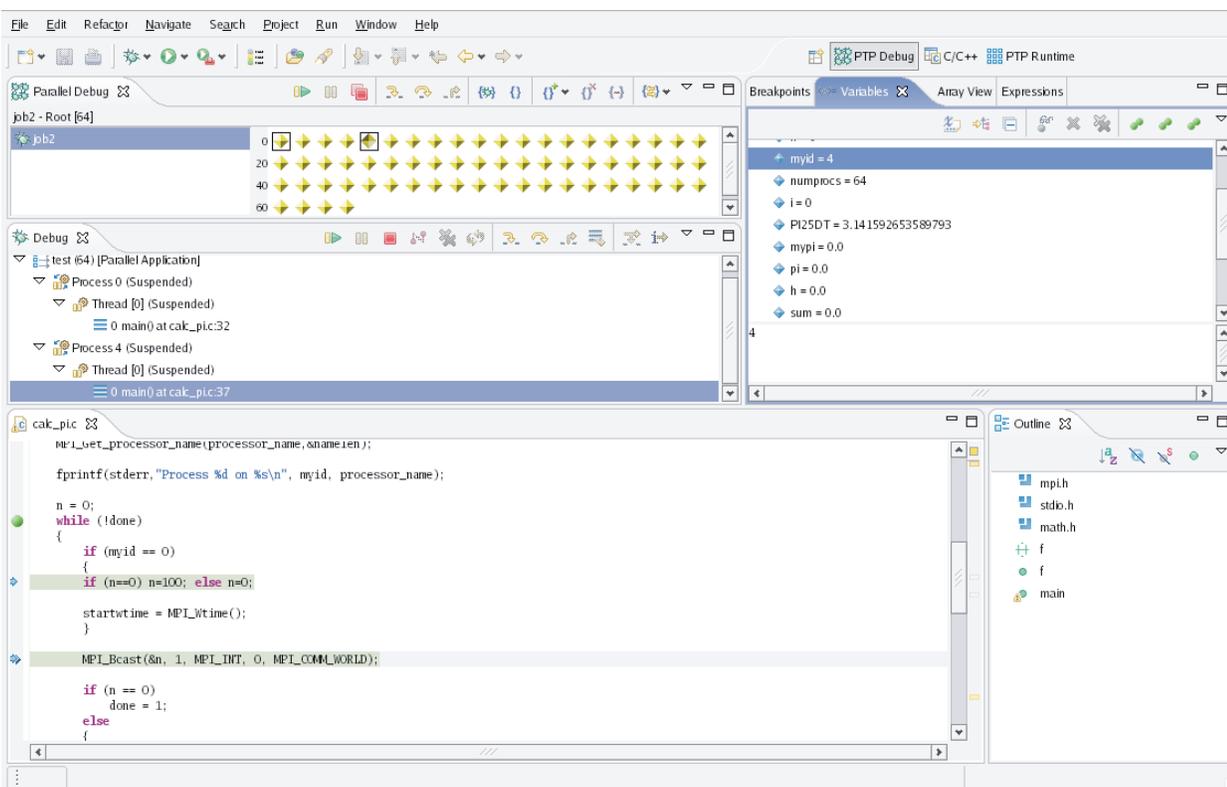
*For more information contact Craig E. Rasmussen at [rasmussn@lanl.gov](mailto:rasmussn@lanl.gov).*

### **Funding Acknowledgements**

NNSA's Advanced Simulation and Computing (ASC).



**Fig. 1.** PTP runtime views showing the node status of a parallel machine (left side) and the process status of a running parallel job (middle). The output from a process, along with a legend showing the different node and process states is also displayed (right side.)



**Fig. 2.** A typical parallel debug session showing the parallel job and associated process status (top left). Directly underneath is a more detailed view of stack frame information from some of the processes being debugged. The program source code and markers indicating breakpoints and current execution locations are displayed (bottom left). Information about the values of variables in one of the processes is also shown (top right.)