Abstract

Eclipse [1] is a widely used, open source integrated development environment that includes support for C, C++, Fortran, and Python. The Parallel Tools Platform (PTP) [2] extends Eclipse to support development on high performance computers. PTP allows the user to run Eclipse on her laptop, while the code is compiled, run, debugged, and profiled on a remote HPC system. PTP provides development assistance for MPI, OpenMP, and UPC; it allows users to submit jobs to the remote batch system and monitor the job queue. It also provides a visual parallel debugger.

The XSEDE community comprises a large part of PTP’s user base, and we are actively working to make PTP a productive, easy-to-use development environment for the full breadth of XSEDE resources. In this paper, we will describe capabilities we have recently added to PTP to better support XSEDE resources. These capabilities include submission and monitoring of jobs on systems running Sun/Oracle Grid Engine, support for GSI authentication and MyProxy logon, support for environment modules, and integration with compilers from Cray and PGI. We will describe ongoing work and directions for future collaboration, including OpenACC support and parallel debugger integration.

Categories and Subject Descriptors
D.2.6 [Software Engineering]: Programming Environments – integrated environments.

General Terms
Documentation, Performance, Languages.

Keywords
Eclipse; IDEs; integrated development environments; Parallel Tools Platform; programming environments; PTP; XSEDE; High Performance Computing

1. Introduction

Computational science models are becoming increasingly complex, with the most complex models exceeding a million lines of code. As software complexity increases, so does the need for many software engineering tools, including tools for version control, issue tracking, automated testing, reverse engineering, and automated refactoring. However, every new tool a scientific programmer must learn is yet another distraction from his primary goal: science.

Eclipse is a widely used, open source integrated development environment (IDE) that seamlessly integrates this functionality into one single, cohesive, graphical tool. It is most popular for Java development, but it has more recently been extended to support C, C++, UPC, Fortran (77 through 2008), Python, and many other languages. It provides point-and-click access to version control systems (including CVS, Subversion, and Git) and issue tracking software (e.g., Bugzilla). It integrates with virtually all existing build systems (including Make) and provides an easy-to-use, graphical debugger. It also provides code completion, code searching, code templates, automated refactoring, and many other tools to ease the processes of writing and understanding code. In short, it makes a full suite of software engineering tools accessible to scientific programmers while only requiring them to learn one tool: Eclipse.

The Eclipse Parallel Tools Platform (PTP) extends the base C/C++/UPC/Fortran capabilities in a number of key directions needed to support scientific application development on high performance computers. First, PTP allows the user to run Eclipse on her laptop, while the code is compiled, run, debugged, and profiled on a remote HPC system. PTP provides integrated...
documentation, code templates, and static analysis tools for MPI, OpenMP, UPC, and other HPC APIs. For code execution, the Parallel Tools Platform provides a graphical interface allowing users to submit jobs to the remote batch system and monitor the job queue. PTP also includes a facility for integrating external performance tools, such as TAU [3], and a visual parallel debugger.

Many of PTP’s users are members of the XSEDE community, and we are actively adding functionality to PTP to make it a viable development environment for XSEDE users. In this paper, we will describe capabilities we have recently added to PTP to support the breadth of XSEDE resources. These include the ability to submit and monitor jobs on systems running Grid Engine, support for GSI authentication and MyProxy logon, support for environment modules, and integration with C/C++ and Fortran compilers from Cray and PGI. We will describe ongoing work and directions for future collaboration, including development assistance features for OpenACC, as well as modifications to broaden the applicability of the PTP parallel debugger. Finally, we will describe how the functionality provided by Eclipse and PTP can facilitate better software engineering practices in the HPC community.

2. THE ECLIPSE PARALLEL TOOLS PLATFORM (PTP)

PTP is based on the Eclipse IDE, which has been successfully used by many developers, using many programming languages, to aid in all aspects of the software development process. Eclipse has many facets. First, it is a vendor neutral, multilingual, open source workbench for software development. It is also an extensible platform for integrating tools, both internal and external to Eclipse. Finally, it is based on a plug-in framework that one can use to create, integrate, and utilize software tools.

PTP was developed to support all aspects of science and engineering application development, from coding and analysis, to launching and monitoring (on remote HPC resources), to debugging and performance tuning. It supports a wide range of parallel programming technologies (including MPI, OpenMP, and UPC). It also supports numerous runtime systems, including batch resource managers such as TORQUE and MPI runtimes such as OpenMPI. It includes a framework for integrating external parallel tools, such as the TAU Performance System [3], including integration of feedback from these tools. PTP has been designed to help simplify users’ interactions with the diverse range of HPC systems that they will likely encounter as they develop, port, debug and optimize their application codes. Besides being able to integrate external tools, PTP comes preconfigured to recognize errors and warnings from popular compilers, integrating the reported problems with the source code view. This facility can also be used to integrate compiler feedback (e.g., reports about successful or unsuccessful loop vectorization) directly into the source view in Eclipse.

PTP includes configurations for popular batch and MPI runtime systems, and can be further customized for new runtime systems or site policies through a comprehensive XML configuration document. Scalable monitoring of the remote HPC resources is also provided as part of the runtime support.

Besides the traditional “local” project type supported by Eclipse, which allows users to develop applications on local resources (such as their laptop), PTP supports a fully remote project type, in which source, builds and runs occur on a remote system, as well as a “synchronized” project type, in which source code exists not only on the developer’s laptop but also on any number of remote HPC resources. A developer can set up any number of build configurations to support builds on the remote systems. Synchronization of source code occurs no less frequently than at build time.

Eclipse’s “Team” support allows projects to be connected to version control systems, including CVS, Subversion, and Git. Team support is extended by a component called the Mylyn Bridge, which links source code in a repository with issue tracking systems such as Bugzilla and Jira. This allows the developer to fully associate issues with the source, as checked into the repository, to help the engineer be more productive in her work. Eclipse also has a hierarchical, indexed help system, which is accessible within Eclipse as well as exportable to a web server, for convenient access even from outside the Eclipse tools environment. Help documentation is primarily written in HTML,
and it is straightforward to integrate documentation produced by tools such as Doxygen into this system, as well as hand-crafted help for one’s application.

3. EXTENDING PTP FOR XSEDE

At NCSA, we are actively working to make Eclipse a viable alternative to the command line for simple application development on XSEDE resources. Users should be able to import source code from a version control system, edit the code, compile it using any of the available compilers on the remote system, and then run the application by submitting and monitoring a batch job, all from within Eclipse. The 2011 release of Eclipse and PTP (a.k.a. the “Indigo” release) contains most of the infrastructure needed to make this possible. Indeed, it allows users to perform these tasks quite well on typical Linux clusters. However, we have found a number of customizations and feature additions necessary to make the same possible for the full breadth and diversity of XSEDE resources. While PTP does not yet fully support every XSEDE system, our recent work on PTP has made great strides toward realizing this vision.

Our recent additions to PTP focused on five areas: job submission and monitoring, supporting environment modules, supporting GSI authentication and MyProxy logon, increasing the number of C/C++ and Fortran compilers supported in Eclipse, and improving Eclipse’s developer assistance features for MPI, OpenMP, and OpenACC.

3.1 Job Submission

In a command-line environment, system administrators typically provide sample job scripts, which users use as templates in order to submit their own jobs to the batch scheduler. In PTP, it is possible to write and submit custom job scripts, exactly as one would do on the command line, but PTP provides another means to submit jobs, which is much simpler.

When the user uses this simpler job submission interface, he never sees a job script. Instead, a dialog box allows him to set job attributes (e.g., the batch queue, wall clock limit, number of MPI processes, and number of processes per node) and submit the job to the batch scheduler. All of this is done using a graphical interface.

This graphical interface has the added advantage that it can be customized for particular machines. Figure 1 (a) shows the customized interface for submitting jobs to Lonestar (a 1,888-node Linux cluster located at the Texas Advanced Computing Center). The set of options is fairly typical for a Grid Engine job submission. However, the descriptive text is customized for Lonestar, as are many of the options and defaults. For example, the default launch command is ibrun rather than mpirun, and the list of parallel environments is Lonestar-specific.

Customizing the job submission dialog for individual machines is possible—and highly recommended—since it is defined entirely using XML. PTP includes generic XML configurations for TORQUE/PBS, Grid Engine, OpenMPI, and SLURM, as well as machine-specific, customized XML configurations for Forge, Kraken, Lonestar, and Ranger. Users can use these “out of the box,” or they can use them as templates to build their own.

3.2 Job Monitoring

PTP also provides a graphical interface for job monitoring, based on LLview [4,11]. It supports systems running TORQUE/PBS, as well as Grid Engine, IBM Parallel Environment, LoadLeveler, OpenMPI, and SLURM, as well as Cray systems running TORQUE with ALPS.

Figure 1(b) shows this monitoring interface running on the CrayXE6/XK6 Blue Waters Early Science System at NCSA. On the left is a list of jobs submitted to the batch system, with information similar to that provided by qstat. (From this list, users can delete their jobs or view their output via a context menu.) On the right is a visualization of the nodes in the machine, organized into rows and columns in a fashion that mimics the machine’s physical organization into cabinets, chassis, and blades. Jobs are colored identically in the job list and the machine visualization, allowing the user to see which nodes a particular job is running on (and which nodes are free).

3.3 Modules Support

Environment modules [5, 6] allow users to switch between different version of compilers, libraries, and other installed software. Modules are widely used in HPC. They are especially on Cray systems: when a user wants to compile a program on a Cray, she generally does not invoke the compiler directly but rather loads a module for a particular programming environment (e.g., module swap PrgEnv-cray PrgEnv-gnu) and then invokes a generic compiler driver (e.g., cc or ftn), which delegates to the compiler for the currently loaded programming environment.

Figure 2. Configuring environment modules for build.

In Eclipse, when a user wants to compile C/C++ or Fortran code, Eclipse simply runs make (typically on the login node of the HPC system). We added support to PTP allowing the user to configure a set of environment modules that will be loaded at build time, prior to running make. The user interface is shown in Figure 2. A list of available environment modules is shown to the user, who indicates which modules are to be loaded. A “Select Defaults” button resets the selection to the modules present in a new login shell. Alternatively, the user can bypass the list and manually enter a list of module commands.

3.4 GSI Authentication and MyProxy Logon

The Grid Security Infrastructure (GSI) provides a single sign-on capability, which allows users to authenticate to a MyProxy server and then access XSEDE resources without the need to re-enter login information for the next several hours. From the command line, this is done using two commands. First, a user runs myproxy-logon to retrieve a short-lived (typically, 12-hour) credential. Then, to gain SSH access to a remote machine, the user runs gssish. The credential is used to authenticate the
When PTP connects to remote machines, it uses an SSH implementation built into Eclipse (namely, JSch [10]). However, this SSH implementation does not support GSI authentication, which prevents users from taking advantage of this single sign-on capability from within Eclipse.

To rectify this, we have created Eclipse plug-ins that add GSI authentication support to Eclipse’s SSH implementation, as well as adding a MyProxy Logon capability to the Eclipse user interface (shown in Figure 3). Once these plug-ins are installed, all of Eclipse’s SSH connections (including connections for remote project synchronization, job launching, and job monitoring) will attempt to authenticate using GSI.

3.5 Broader Compiler Support

When a project is built, the entire output of make is present in Eclipse’s Console view. However, compiler errors and warnings are also displayed in the Problems view, and markers are placed next to the corresponding lines of source code in the editor.

3.6 Broader Programming Model Support

One of Eclipse’s greatest strengths is its ability to assist developers with coding. It provides a syntax-highlighting text editor, of course, but this editor is integrated with other language-aware facilities that can respond and interact with the user, in real time, as he codes. For example, Eclipse provides language-aware auto-completion (also called content assistance) for C/C++ and Fortran. Eclipse’s Outline view displays the high-level structures (classes, functions, etc.) in the current source file; this view is updated in real time as the user edits the file. When the caret is moved over a CALL statement in a Fortran program, the Fortran Declaration view displays the declaration for that function (including any leading comments), even if it is located in another file. Similar information is displayed in a hover tip when the user moves the mouse cursor over an identifier.

PTP extends these developer assistance features with support for MPI, OpenMP, and other parallel programming models. For example, it provides auto-completion, documentation/online help, and hover tips for MPI functions. Code templates are supplied to conveniently add blocks of code for frequently used idioms.

Our recent efforts have focused on improving the integration of developer assistance features in three key areas.

- **Fortran support.** Developer assistance features have been available for C/C++ for several releases. The June 2012 release will make more of them available for Fortran. For example, when the editor caret is moved over CALL statement referencing an MPI procedure, an extensive description of that procedure (based on its OpenMPI man page) is displayed in the Fortran Declaration view.

- **OpenACC support.** We have also focused on extending the existing MPI and OpenMP developer assistance features to support OpenACC [7], a relatively recent standard that allows accelerator devices (e.g., GPUs) to be programmed using OpenMP-style directives. (OpenACC was developed by members of the OpenMP Working Group on Accelerators, and it is expected that it will eventually be merged into OpenMP [8]. It is currently supported (at least partially) in Cray
Additionally, the TAU workflow allows automatic uploading of counter events and automatic selective instrumentation.

- **MPI 2 support.** The upcoming release will include developer assistance features for the entire MPI 2.2 API, including APIs for process creation/management and parallel I/O. (Previous releases supported MPI 1.1.)

Performance profile and trace data, selection of PAPI hardware provided by the ETFw workflow includes the generation of performance analysis tools into Eclipse [3, 9]. TAU support in this way can be browsed and analyzed from within Eclipse. Similar functionality is available for other performance analysis tools which provide their own ETFw workflow definitions.

### 3.8 XSEDE-Specific Customizations

At NCSA, we are also developing an Eclipse plug-in that will be used exclusively by XSEDE users. Its interface is shown in Figure 6. It is simple but useful: it provides menu items to configure PTP for particular XSEDE resources, as well as to open an (Eclipse-integrated) SSH terminal on those resources. Of course, this can be done without the XSEDE plug-in; the plug-in simply expedites the process and alleviates the need to remember hostnames and other details about particular resources. The XSEDE plug-in also provides one-click access to the XSEDE User Portal, as well as direct links to online documentation for individual XSEDE resources. These open in a Web browser inside Eclipse, so the user does not need to switch between Eclipse and an external Web browser.

![Figure 6. XSEDE integration menu.](image)

### 3.7 External Development Tool Support

Early in the development of the PTP, it was determined that switching between Eclipse and the command line to perform common development tasks was not acceptable: HPC developers should be able to perform their entire workflow within Eclipse. However, many of the tools used by HPC developers, including performance and memory analysis applications, are intended for use on the command line. The number of such tools made developing Eclipse-based replacements infeasible. Instead, we augmented PTP with the ability to wrap command-line based utilities in its UI and invoke them as part of a user- or developer-defined workflow. This functionality is provided by a component of PTP called the External Tools Framework (ETFw).

An ETFw workflow is defined by an XML document which specifies the commands and operations to be used at three predefined steps in the development process: compilation, execution, and analysis. Depending on the components required by the tool, this can include replacing the default compiler with an alternate or a compiler wrapper, wrapping the executable with a data collection tool, or running an analysis tool on data generated during the application’s execution. Each workflow component can have a UI defined in the XML document, allowing the user to control its behavior from the Eclipse launch configuration interface. Once an XML definition has been created and loaded into Eclipse, it is essentially a one-click operation for the user to generate performance data or accomplish whatever task the workflow defines.

The ETFw arose from preliminary efforts to integrate the TAU performance analysis tools into Eclipse [3, 9]. TAU support provided by the ETFw workflow includes the generation of performance profile and trace data, selection of PAPI hardware counter events and automatic selective instrumentation. Additionally, the TAU workflow allows automatic uploading of profile data to a performance database. Performance data stored in this way can be browsed and analyzed from within Eclipse. Similar functionality is available for other performance analysis tools which provide their own ETFw workflow definitions.

### 3.9 Future Directions

The customizations listed thus far will be publicly available in the June 2012 ("Juno") release of PTP. (The plug-ins for GSI authentication/MyProxy logon and XSEDE customization are not part of PTP itself; they will be distributed separately by NCSA.) These improvements represent a significant step toward providing a comprehensive, robust development environment for XSEDE resources. Of course, more work is still needed in a number of areas, including the following.

#### 3.9.1 Broader Resource Manager Support

PTP currently ships with customized resource managers for Forge, Kraken, Lonestar, and Ranger. However, these represent only a fraction of the available XSEDE resources. Besides the ability to provide "generic" resource manager support for default distributions of say Torque/PBS or SLURM, it is possible to create site-specific resource manager configurations which encode site policies, to ensure that users use the resources in the most optimal and efficient way possible. This is an area of potential collaboration between XSEDE Service Providers and our PTP development efforts. In addition to developing custom configurations, we can work together to develop efficient ways to distribute these configurations to end users of Eclipse and XSEDE Digital Services.

**Case Study: Supporting Blue Gene P/Q.** The Resource Management framework within Eclipse PTP makes it easy (although admittedly nontrivial) to create custom resource managers for new systems. We recently developed an extension for Eclipse PTP to support the IBM BG/P and emerging BG/Q systems at Argonne National Laboratory, which use a customized combination of TORQUE/PBS and COBALT commands for getting partition information. Although there are currently no Blue Gene resources supported by XSEDE, the uniqueness of these systems serves as an interesting case study for PTP’s ability...
to support new systems, and it provides compelling evidence that PTP will be able to integrate with new XSEDE resources over time.

On BG/P, PBS utilities such as `qstat` and `qdel` perform as expected, but the `qsub` command is significantly different. The modified version has fewer parameters, in particular those that specify the hardware requirements of the job. On BG/P, the compute nodes have identical hardware, so those options would be superfluous. In addition, many of the other `qsub` parameters have slightly different syntax. Also, Eclipse PTP typically constructs a job script in order to submit jobs to a batch scheduler, whereas on the BG/P the `qsub` parameters are typically passed in on the command line (although a script option is available).

In order to get hardware information about the system, the BG/P systems at the Argonne Leadership Computing Facility (ALCF) use the command `partlist`, which returns a list of hardware partitions. These partitions are not mutually exclusive, and in fact overlap in many different ways. However, the partitions define pre-configured blocks of contiguous computation nodes in order to configure an ideal network topology between nodes with respect to connection latency. One task in writing the extension was mapping the partition names to hardware locations. In the BG architectures, systems are organized in a row / rack / midplane / node-card / compute-card / chip hierarchy. Because of a well-defined naming convention for the partitions, mapping the partitions to hardware locations is straightforward. This process is done both for generating the System Monitoring perspective in Eclipse PTP and also in mapping currently running jobs to the hardware locations. The BG/Q test system names the partitions using 5-dimensional torus coordinates, but there exists a straightforward mapping of those coordinates to hardware locations.

Finally, it should be noted that because of the modular and flexible nature of the Eclipse PTP Resource Manager design, no source code on the client side of Eclipse PTP had to be modified in order to add support. On the client side, there is an XML file that defines the configuration for interacting with the `qsub` and `qstat` commands, and for building the user interface. On the server side, the Perl scripts that get job and system information for the System Manager view had to be modified to call the appropriate utility and parse the output. The only possible complication would have been the hardware view in the System Monitor perspective, but BG support had previously been added to Eclipse PTP. Figure 7 shows the System Monitor view in Eclipse PTP, populated with the prototype support.

3.9.2 Parallel Debugger Integration

PTP includes a parallel debugger, which is designed for debugging MPI programs on clusters. Currently, the debugger only works on a limited number of XSEDE resources (notably, NCSA’s Forge). We are working to improve both the back-end scalability of the PTP parallel debugger as well as the scalability of the display. We are also investigating ways to easily manage the deployment of the back-end infrastructure for the debugger.

4. ENABLING SOFTWARE ENGINEERING

Eclipse PTP is able to make many software engineering tasks considerably easier or even feasible. For starters, one significant advantage of PTP, indeed, any IDE, is the visibility one gains into the source structure and content, such as seen in Figure 8 for the application HOMME.

![Figure 7. Blue Gene/Q System Monitoring Perspective, showing top level of architectural hierarchy.](image)

Figure 7. Blue Gene/Q System Monitoring Perspective, showing top level of architectural hierarchy.

Another aspect of assistance for the software engineer is to help deduce the call hierarchy of a new application. In Figure 9, one can see a flat source directory hierarchy, and a main function in the source file `startup.c`. Using the call hierarchy capability in the C/C++ Development Toolkit which is part of PTP, one can quickly gain an understanding of the call hierarchy based on static analysis, as seen in Figure 10.

Application software engineers are often faced with the challenge of managing code development amongst a variety of HPC resources. PTP offers the capability of a synchronized remote project, in which the uncommitted master source code is held locally at a developer’s workstation, with source synchronization to one or more HPC resources. Figure 11 shows a schematic of how this service functions.

One important aspect of software engineering for science and engineering applications is performance tuning. Eclipse PTP provides an integration point for popular performance analysis tools such as TAU, through its External Tools Framework (ETFw). Besides providing support for launching performance experiments, Eclipse provides mechanisms for linking data with visualization tools as well as direct feedback mechanisms to indicate performance issues directly in one’s source code.

![Figure 8. Source navigation for the application HOMME: directory structure (left), syntax-aware editor (middle), and code outline (right), in the Fortran perspective.](image)

Figure 8. Source navigation for the application HOMME: directory structure (left), syntax-aware editor (middle), and code outline (right), in the Fortran perspective.
Eclipse PTP also takes advantage of comprehensive support for source code repositories, through the Team support features in Eclipse. Repositories supported include CVS, Subversion, Git, and more; common features amongst the repositories include source version numbers and their commit comments/descriptions, indications of source files which have uncommitted changes, as well as menus for driving many interactions with the repository. Finally, Team support provides visual mechanisms for comparing different versions of a source, allowing the developer easy means to select which changes they want to preserve in a source code merge operation.

Besides supporting source repositories, which are essential for collaborative code development, one can use a capability bundled in Eclipse for integration of source code with issue tracking. This capability, called the Mylyn Bridge, allows a software engineer to track issues, in the context of her source code, as presented in Eclipse. Context is preserved between the issue tracking system (such as Bugzilla) and their source repository, and presented in the context of the source code navigation views built into Eclipse.

Finally, Eclipse provides a mechanism to help software engineers provide documentation for their project (usually an engineer’s least favorite task). The Eclipse Help System, as seen in Figure 10, provides a searchable, hierarchical documentation system that is remarkably easy to develop collaboratively. Additionally, plug-ins are available to ease development of the HTML documentation. The help system is then deployable within the context of a normal Apache httpd server, which provides opportunities for wider dissemination of the documentation and its usage outside Eclipse.

5. CONCLUSIONS

Eclipse provides a sophisticated, intelligent, and extensible software development environment. Source code navigation features aid in reverse engineering and software maintenance. Integration with version control and issue tracking systems support best practices in software configuration management. A searchable, independently deployable online help system is ideal for providing user and developer documentation. PTP extends Eclipse with capabilities that can substantially ease the process of building HPC applications: code editing and navigation, compilation, job submission and monitoring, debugging, and performance tuning are all integrated into a single tool. With the support of the XSEDE community, we believe that PTP has the potential to become a full-featured development environment for XSEDE. More information on PTP can be found at http://www.eclipse.org/ptp.

6. ACKNOWLEDGMENT

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (award number OCI 07-25070), the Workbench for High Performance Computing (WHPC) SI2 project, which is supported by the National Science Foundation (award number OCI-1047956), and the state of Illinois. Blue Waters is a joint
effort of the University of Illinois at Urbana-Champaign, its National Center for Supercomputing Applications, Cray, and the Great Lakes Consortium for Petascale Computation. Additionally, portions of this material are supported by or based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under its Agreement No. HR0011-07-9-0002.

7. REFERENCES


