The Eclipse Parallel Tools Platform

Toward an Integrated Development Environment for Improved Software Engineering on Crays

Jay Alameda and Jeffrey L. Overbey
National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign
Urbana, IL, USA
{alameda,overbey2}@illinois.edu

***Abstract*—Eclipse is a widely used, open source integrated development environment that includes support for C, C++, Fortran, and Python. The Parallel Tools Platform (PTP) 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; and it provides a visual parallel debugger.**

In this paper, we will describe the capabilities we have added to PTP to support Blue Waters, the Cray XE6/XK6 system being installed at NCSA. These capabilities include submission and monitoring of ALPS jobs, support for OpenACC, and integration with Cray compilers. We will describe ongoing work and directions for future collaboration, including integration with CrayPat, Loopmark compiler feedback, and parallel debugger integration.

Keywords-Blue Waters; Eclipse; IDEs; integrated development environments; Parallel Tools Platform; programming environments; PTP

I. 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 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.

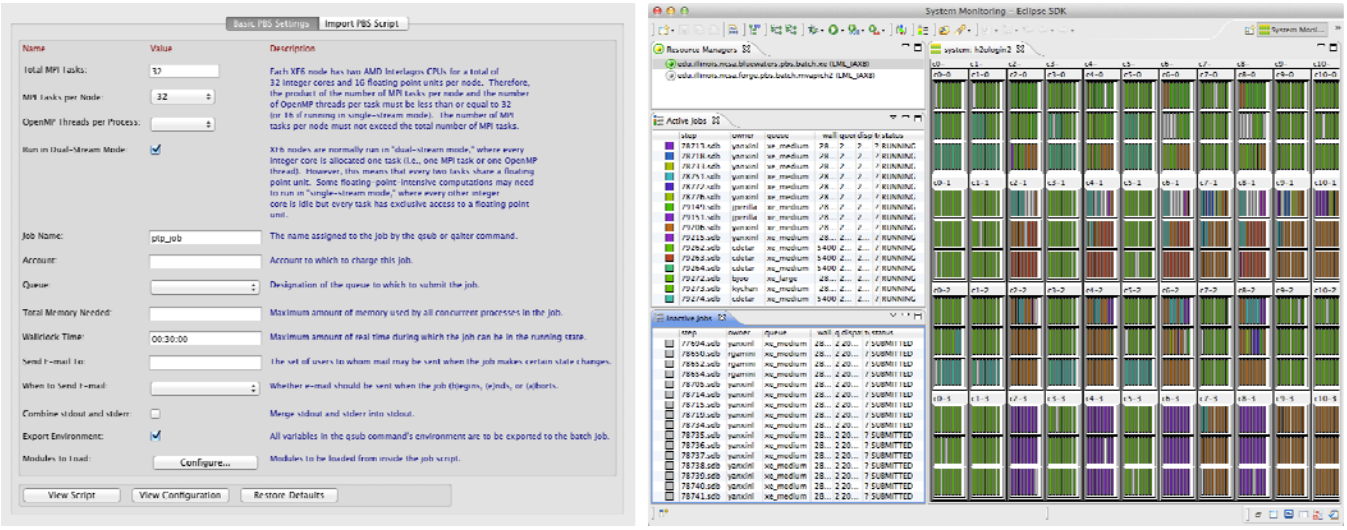
The Eclipse Parallel Tools Platform (PTP) extends the base C/C++/UPC/Fortran capabilities in a number of key directions needed to support scientific application development on high performance computers. First, PTP allows the user to run Eclipse on her laptop, while the code is compiled, run, debugged, and profiled on a remote HPC system. PTP also provides development assistance for MPI, OpenMP, and UPC, including integrated documentation and static analysis tools. 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, and a visual parallel debugger.

For the past several months, we have been extending the Eclipse Parallel Tools Platform to work with Blue Waters, the Cray XE6/XK6 system being installed at the National Center for Supercomputing Applications at the University of Illinois. In this paper, we will describe the capabilities we have added to PTP to support Cray machines, including submission and monitoring of ALPS jobs, support for OpenACC, and integration with Cray compilers. We will describe ongoing work and directions for future collaboration, including integration with CrayPat, Loopmark compiler feedback, and parallel debugger integration.

II. 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



(a) (b)

Figure 1. PTP's graphical user interface for (a) submitting and (b) monitoring jobs on the Blue Waters prototype system.

resources), to debugging and performance tuning. The foundational IDE in PTP 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, including integration of feedback from these tools. PTP has been designed to help simplify a 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.

The support for runtime systems is delivered with many 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, in which one can use Eclipse to develop applications on their local resource (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 a developer's source code exists not only on their laptop computer, 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 their work. Eclipse also has a hierarchical, indexed help system, which is accessible within Eclipse as well as exportable to a web server. 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.

III. Extending PTP for Blue Waters

At NCSA, we are actively working to make Eclipse a viable alternative to the command line for simple application development on Blue Waters. Users should be able to import source code from a version control system, edit the code, compile it using the Cray, GNU or PGI compilers, and then run the application by submitting and monitoring a batch job, all from within Eclipse. The 2011 release of Eclipse and PTP (the so-called "Indigo" release) had most of the infrastructure needed to make this possible—indeed, users could perform these tasks quite well on typical Linux clusters—but we had to make a number of additions to make the same possible on a Cray.

Our additions to PTP focused on four areas: customizing the user interface for job submission and monitoring, supporting environment modules, supporting the C/C++ and Fortran compilers from Cray and PGI (GNU was already supported), and supporting OpenACC development.

A. *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. The customized interface for submitting jobs to Blue Waters is shown in **Figure 1**(a). Note the extensive explanatory text, as well as

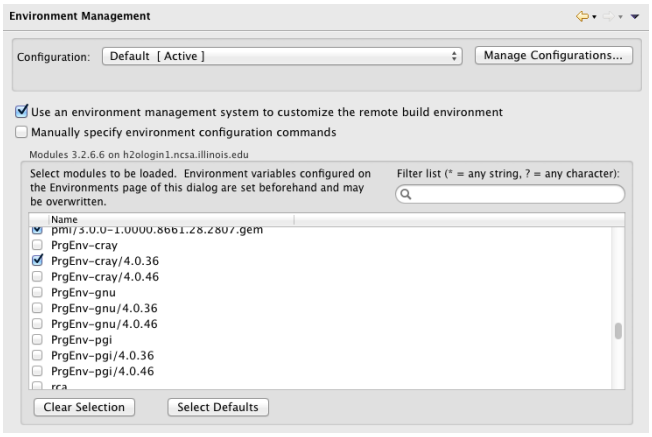


Figure 2. Configuring environment modules for build.

the "Run in Dual-Stream Mode" checkbox (an `aprun` option for the Cray XE6 that has no equivalent on a typical `mpirun`).

B. *Job Monitoring*

PTP also provides a graphical interface for job monitoring, based on LLview [6]. Although TORQUE has been supported for several years, our colleagues at Jülich Supercomputing Centre recently added support for job monitoring on Crays, which internally uses a combination of TORQUE and ALPS commands.

Figure 1(b) shows this monitoring interface running on the Blue Waters Early Science System. 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 (roughly similar the allocation grid displayed by `xtnodestat`). Jobs are colored identically in the jobs list and the machine visualization, allowing the user to see which nodes a particular job is running on (and which nodes are free).

C. *Modules Support*

Environment modules [7][8] allow users to switch between different version of compilers, libraries, and other installed software. Modules are common in HPC systems, although they are especially important on Cray systems. When a user wants to compile a program on a Cray, she generally does not invoke the compiler directly; rather, she 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.

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). On Cray systems, the Makefile will typically invoke the `cc` or `ftn` drivers,

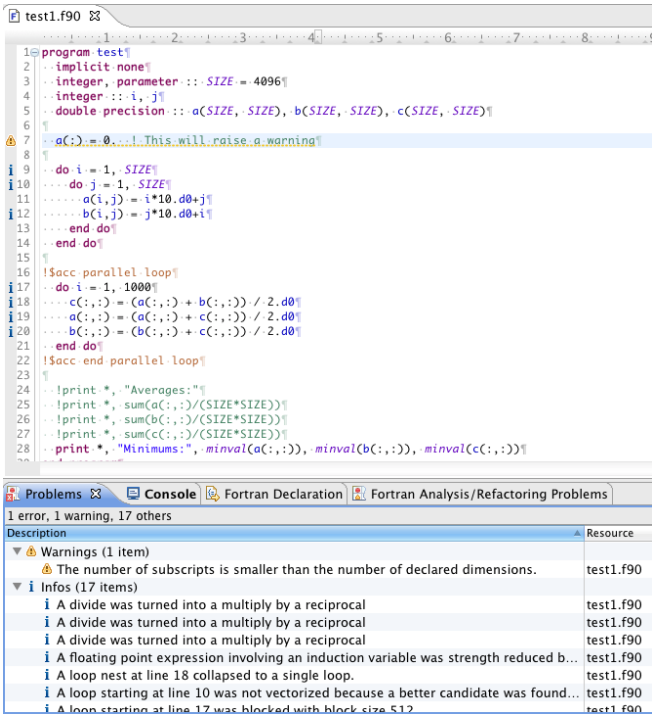


Figure 3. Recognition of output from the Cray Fortran compiler.

assuming that modules for the appropriate programming environment have been loaded.

Thus, 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.

D. *Cray and PGI 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. **Figure 3** shows an example: the Cray Fortran compiler indicated a warning on line 7, and Eclipse has accordingly placed a warning marker next to line 7 in the left margin of the source editor.

Internally, marking errors and warnings requires that Eclipse be able to "understand" the compiler's output messages, at least well enough to identify the message text

and associate it with the correct source file and line number. Previous releases of PTP could recognize messages produced by a number of compilers (including the GNU compilers); in recent months, we added support for the Cray, PGI, and Open64 compilers for C, C++, and Fortran.

Figure 3 shows Eclipse recognizing a warning from the Cray Fortran compiler. In addition, the Cray compiler can also produce information about what optimizations were (not) performed (via the `-h` command line switch); if these are found in the output of `make`, they will be displayed in the Problems view under the "Info" category, and the corresponding source lines marked accordingly.

E. OpenACC Support

OpenACC [9] is 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 the OpenACC API will eventually be merged into OpenMP [10]. OpenACC is currently supported (at least partially) in Cray and PGI's compilers.

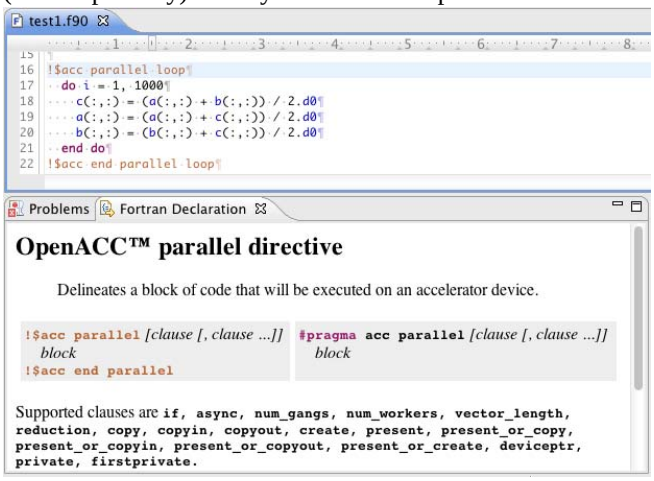


Figure 4. Integrated OpenACC Documentation

In Eclipse, we have added user assistance features for OpenACC in two key areas.

- **Documentation.** In the Fortran editor, when the Fortran Declaration view is enabled, this view will display a brief summary when the editor caret is placed on an OpenACC directive or procedure call. This is shown in **Figure 4**. (Similar information is available in the C/C++ editor.)
- **Content Assist.** In the Fortran editor, if the user types part of the `$!acc` directive prefix and presses Ctrl+Space, a pop-up list of OpenACC directives is shown to the user (as shown in **Figure 5**); this list is refined as the user types more characters, and pressing Enter inserts the selected directive's text. (This facility is called *content assist* in Eclipse; it is more generally called *auto-completion* and is similar to Visual Studio's IntelliSense.) After the directive is inserted, the integrated OpenACC documentation is displayed (as described previously), informing the user of the supported clauses.

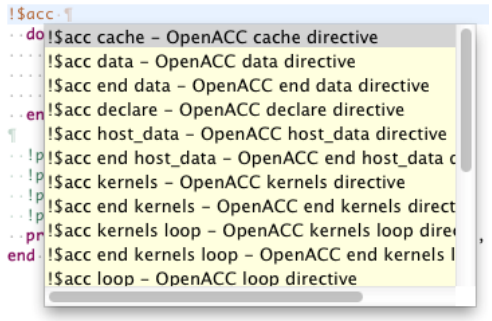


Figure 5. Content assistance/templates for OpenACC directives.

F. Future Directions

The customizations listed thus far will be publicly available in the June 2012 ("Juno") release of Eclipse. These represent, the most critical changes needed to make development on Blue Waters possible given the short period of time between the Blue Waters platform change and the Juno release in June 2012. Providing a comprehensive, robust development environment for Crays will require research and development in a number of areas, including the following.

- **Parallel Debugger Integration.** PTP includes a parallel debugger, which is designed for debugging MPI programs on clusters. With the appropriate ALPS integration, it may be possible to use this debugger on Crays, at least for small- to medium-scale debugging.
- **Craypat Integration.** PTP includes a component called the External Tools Framework, or ETFw, which allows command-line tools to be launched from Eclipse. It allows the Eclipse GUI to solicit arguments from the user, run the external tool, and then launch any analysis of the results. ETFw has been used to integrate the TAU Performance System [5] with Eclipse and could likewise be used to integrate the Cray Performance Analysis Tools.
- **Improved Loopmark Integration.** There is another component of ETFw called the Feedback View. An external tool can write an XML file containing information about a source file (e.g., what compiler optimizations were, or were not, performed); then, the Feedback View will display this information in a table, as well as place visual markers on the corresponding lines of source code. This could be an alternative means of integrating the Cray compiler's loopmark information into Eclipse.
- **Refactorings for OpenACC.** A refactoring tool automates certain, methodical changes to source code while ensuring that they will not change the program's externally observable behavior. A typical example is *Rename*, a refactoring that changes the name of a procedure and updates all of its call sites

accordingly (and fails if the new name is invalid). Photran includes extensive automated refactoring support [1][2][3][4]. A number of new refactorings could be added to support OpenACC development. One example is *Make Loop OpenACC-Parallel:* the user would select a DO-loop; Photran would check the context and contents of the loop, ensuring that introducing an OpenACC *parallel loop* directive would not change the program's behavior; then, if it is determined to be safe, Photran would insert the surrounding OpenACC *parallel loop/end parallel loop* directives.

IV. 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 6** for the application HOMME.

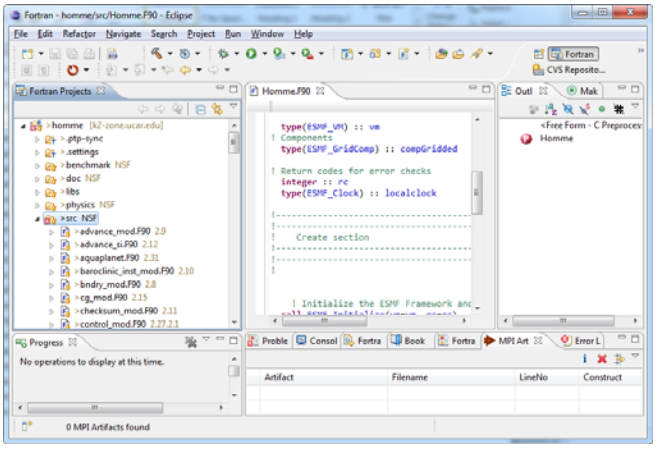


Figure 6 Source navigation for the application HOMME: directory structure (left), syntax-aware editor (middle), and code outline (right), in the Fortran perspective.

Another aspect of assistance for the software engineer is to help deduce the call hierarchy of a new application. In **Figure 7**, 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 8**.

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 9** 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

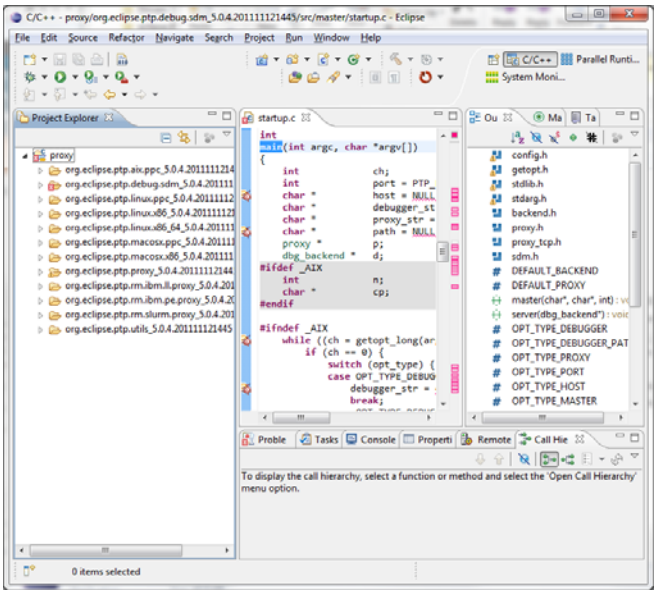


Figure 7. Source distribution with few hints as to the logical structure of the source code.

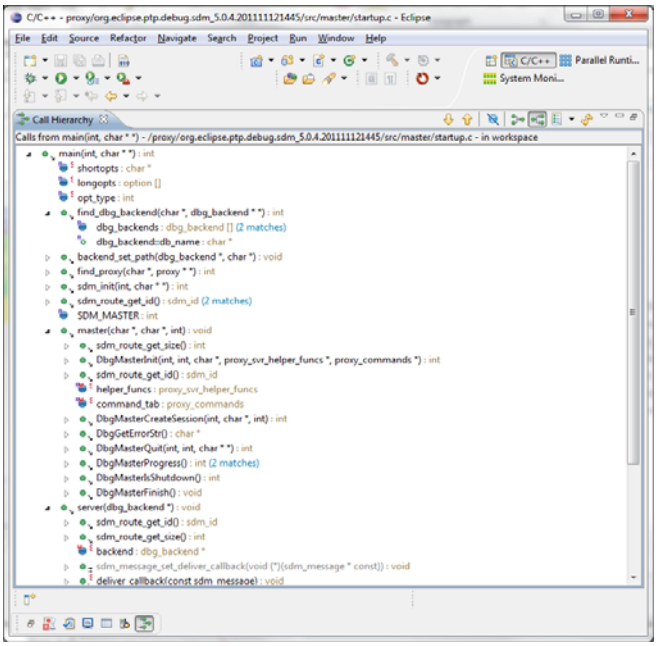


Figure 8. Navigable call hierarchy derived from the `main` function in *startup.c*.

issues directly in one's source code.

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, 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

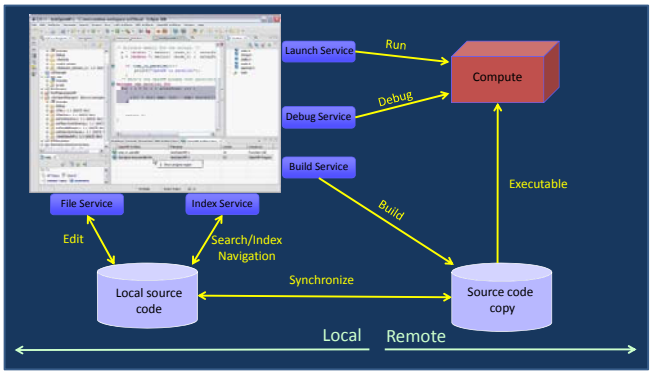


Figure 9. Schematic of synchronized project capability in PTP. Note that build, run and debug occurs on the remote HPC system.

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 source code-issue tracking integration. This capability, called the Mylyn Bridge, allows a software engineer to track issues, in the context of their 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

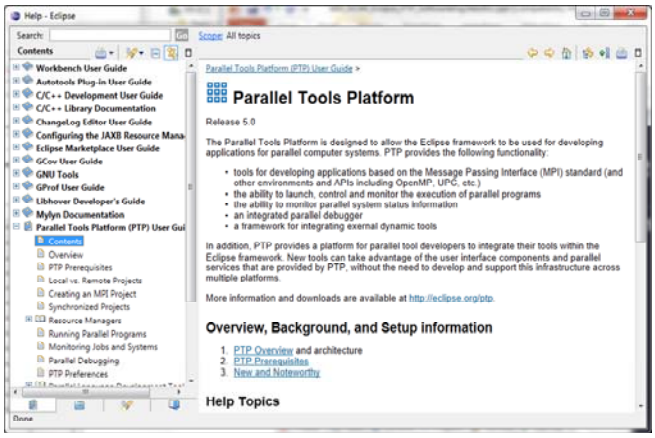


Figure 10. The Eclipse Help System.

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 one's documentation.

V. 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. Our work supporting Blue Waters in PTP is a first step toward making this environment available to Cray developers. With the support of the Cray community, we believe that PTP has the potential to become a full-featured development environment for Cray application developers. More information on PTP can be found at http://www.eclipse.org/ptp.

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.

References

[1] J. Overbey, S. Xanthos, R. Johnson, and B. Foote, "Refactorings for Fortran and High-Performance Computing," 2nd Intl. Workshop Soft. Eng. for High Performance Computing System Applications (SE-HPCS 05).

[2] J. Overbey, S. Negara, and R. Johnson, "Refactoring and the Evolution of Fortran," 2nd Intl. Workshop Soft. Eng. for Computational Science and Engineering (SECSE 09).

[3] M. Méndez, J. Overbey, A. Garrido, F. Tinetti, and R. Johnson, "A Catalog and Classification of Fortran Refactorings," 11th Argentine Symp. Soft. Eng. (ASSE 2010).

[4] J. Overbey, M. Fotzler, A. Kasza, and R. Johnson, "A Collection of Refactoring Specifications for Fortran 95," ACM Fortran Forum, vol. 29, issue 3, pp. 11–25, December 2010.

[5] S. Shende and A. D. Malony, "The TAU Parallel Performance System," International Journal of High Performance Computing Applications, vol. 20, issue 2, pp. 287–331, Summer 2006.

[6] LLview [Online]. Available: http://www2.fz-juelich.de/jsc/llview/

[7] J. L. Furlani and P. W. Osel, "Abstract Yourself with Modules," Proc. 10th Large Installation Sys. Admin. Conf. (LISA 96), pp. 193–204.

[8] Modules – Software Environment Management [Online]. Available: http://modules.sourceforge.net/

[9] OpenACC [Online]. Available: http://www.openacc-standard.org/

[10] OpenACC – Frequently Asked Questions [Online]. Available: http://www.openacc-standard.org/Frequently-Asked-Questions