Eclipse-PTP: An Integrated Environment for the Development of Parallel Applications

Greg Watson (grw@us.ibm.com)
Craig Rasmussen (rasmusen@lanl.gov)
Beth Tibbitts (tibbitts@us.ibm.com)

Parallel Tools Workshop, HLRS, July 2008
Contents

- What is Eclipse?
- PTP Overview
- Remote Development
- Demo
- Challenges
- Conclusion
What is Eclipse?

- Cross-platform open source framework for highly integrated state-of-the-art development tools
- Integrated tools include:
  - Project management
  - Advanced editing
  - Automated build system
  - Revision control (CVS, SVN)
  - Visual debugger
  - Much more…
- Designed to be robust, scalable, commercial quality
- Available for Linux, Unix and Windows
- Multi-language support for Java, C, C++, Fortran, Python, Perl, PHP, and others
C/C++/Fortran Features

- C/C++ Development Tools (CDT) adds C and C++ support
- Photran adds Fortran support
- Standard (Makefile) and managed builders
- Visual debugging using GDB
- High level views (outline view, call hierarchy, type hierarchy, include browser)
- Advanced searching (types, functions, variables, declaration, reference, etc.)
- Content assist
- Context sensitive help
- Simple refactorings (rename, extract constant, more coming…)

© 2008 IBM Corporation; made available under the EPL 1.0
What is PTP?

Enabling Eclipse for Parallel Application Development
http://eclipse.org/ptp

Best practice tools for experienced parallel programmers

Leverage Eclipse ecosystem and community for development and support

Parallel Tools Platform

Improve parallel tools and the productivity of tool developers

Provide focal point for parallel tool development for a broad range of architectures

Tools to assist new breed of programmers to develop parallel programs
PTP Goals

- Bring best practice tools to parallel programmers
- Provide tools designed to specifically address parallel programming problems
- Reduce the complexity of developing applications for parallel systems
- An open extensible platform for developing new/advanced tools
  - To address petascale and multicore programming and productivity issues
PTP History

- Eclipse Foundation open-source project
- Part of the Tools top-level project (same as CDT)

Creation
Apr '05

PTP 1.0
Mar '06

Tools Transition
Dec '06

PTP 1.1
Feb '07

PTP 2.0
Mar '08

PTP 2.1
Aug '08

2005  2006  2007  2008  2009
Contributors

- Los Alamos National Laboratory
- IBM
- Oak Ridge National Laboratory
- University of Oregon
- Monash University
- Munich University of Technology
- University of Tennessee
Application Development Cycle

Coding & Analysis Tools

Launching & Monitoring Tools

Performance Tuning Tools

Debugging Tools

eclipse
Coding & Analysis Tools

- Uses Eclipse & CDT tools
  - Project management, editing, advanced searching, refactoring, version control, deployment

- Parallel Language Development Tools (PLDT)
  - Assistance tools
    - Identify MPI/OpenMP/LAPI/UPC “artifacts”
    - Advanced help (hover, content assist, etc.),
    - Wizard for new MPI/OpenMP project creation
  - Static Analysis tools
    - OpenMP concurrency analysis
    - MPI barrier analysis to detect deadlocks
  - Dynamic Tools coming
Parallel Language Development Tools (PLDT)

- **Assistance tools to increase productivity of parallel programmers**
  - New project wizards (MPI, OpenMP)
  - Content Assist (command/API completion), hover help, built-in API help descriptions in an html help "view" (MPI, OpenMP, LAPI, UPC)
  - Location of parallel “artifacts” in code: MPI, OpenMP, LAPI APIs, etc

- **Static analysis tools**
  - MPI Barrier analysis – deadlock detection
  - OpenMP concurrency analysis & problems
Parallel Language Development Tools (PLDT)

- Dynamic Tools
  - Taskfinder

- Command line tool by Christoph von Praun
- Dynamic tool launching from Eclipse/PTP
- Utilizes Performance Tools Framework
- Dynamically samples to propose “Task Heads” – potential areas for parallelization
Launching & Monitoring Tools

- Improves visibility into target system
- Single point of interface for launching and control
- Manages interaction with different runtime systems and job schedulers
Runtime Environment

- Extensible framework for monitoring
  - System and node status
  - Job status (e.g. position in queue)
  - Application status
- Plug-in adapters to support different resource managers
  - Job schedulers (e.g. LoadLeveler)
  - Runtime systems (e.g. PE or OpenMPI)
- Adapters manage
  - Status information
  - Application launch & control
  - Debug launch
- Local or remote monitoring and launching supported
Debugging Tools

- Scalable integrated debugger
  - One-click-debugging™
- Targeted at SPMD programming models
- Supports mixed MPI & thread debugging
- Platform for building new debugging paradigms
Parallel Debugger

- Tightly integrated with Eclipse
- Infrastructure designed for very large applications
  - UI scaling issues still to be addressed
- Language neutral
- Supports debugging multiple jobs simultaneously
- Client/server architecture
- Utilizes backend debugger (e.g. gdb) for low level operations
- Standard debug activities
  - Breakpoints, stepping, viewing variables, etc.
- Group operations
  - Commands can be applied to groups of processes
  - Can query groups of processes for information
Performance Tuning Tools

- Goal: Leverage Eclipse integration for performance instrumentation, analysis, visualization
- A variety of existing tools are available
- PTP Performance Tools Framework integration points:
  - Instrumentation; automatic and selective
  - Build; may be transparent to user
  - Launch with instrumentation
  - Management of profile/trace data
  - Analysis/visualization launch
Performance Tools Framework

- Reduce the “eclipse plumbing” necessary to integrate tools
- Provide integration for instrumentation, measurement, and analysis for a variety of performance tools
  - Dynamic Tool Definitions:
    - Workflows & UI
    - Tools and tool workflows are specified in an XML file
    - Tools are selected and configured by users in the launch configuration window
    - Output is generated, managed and analyzed as specified in the workflow

```xml
<tool name="Valgrind">  
  <execute>  
    <utility command="bash" group="init">  
      <utility command="valgrind" group="valgrind">  
        <optionpane title="Valgrind variables">  
          <top option label="Leak Check" options="--leak-check=full" tooltip="Full memory leak check">  
            <top option label="Show Reachable" options="--show-reachable=yes" tooltip="Show reachable" />  
          </top option>  
        </optionpane>  
      </utility>  
    </utility command="bash" group="init">  
  </execute>  
</tool>
```
Performance Tools Framework Users

- U. Oregon: TAU, ParaProf
  - Utilizes workflow framework
- IBM: Taskfinder integration
  - Dynamic tool launch
- Others interested:
  - IBM: HPC Toolkit, VPA, Tuning Fork
  - UTK: PAPI
Remote Development Environment (RDE)

- Adds transparent remote development capability to CDT
- Builds on changes introduced into CDT 5.0 that allow projects to reside remotely
- Uses PTP service model to allow CDT services (e.g. build, index, launch, debug, etc.) to be independently selected as local or remote
- Adds remote project creation wizard

![Diagram of Remote Development Environment (RDE)]
Remote Tools Framework

- Abstraction layer provides a common interface to arbitrary remote (or local) service providers
  - Connection management
  - Process/script execution
  - File manipulation
  - File/directory browser
- API designed for use by Eclipse-based tools
- Supported service providers
  - Remote tools
  - Remote system explorer (RSE)
  - Local
Demo
PTP Challenge Area: Fortran

- Fortran support is not as robust as CDT
  - Support is critical for PTP adoption by HPC/Govt users

- Photran project
  - Development has almost stopped
  - Tech lead at UIUC, Jeff Overbey, at IBM for summer internship
  - AST, refactoring plans

- Photran features:
  - Supports Fortran 77, 90, and 95
  - Syntax-highlighting editor
  - GUI interface to gdb
  - Makefile-based compilation
  - Compiler error extraction
  - Outline view
  - Open declaration
  - Rename and Introduce Implicit None refactorings

http://eclipse.org/photran
Conclusion

- Production quality?
  - Enhancements in 2.1 release in August to simplify porting to new platforms, improve debugger
  - Fortran work is proceeding slowly, more resources required
  - Additional work required for very large scale systems
- Need “friendly” users willing to provide feedback
- Need more developers to help improve core functionality
- Exciting opportunity for ongoing research
  - Static analysis/refactoring
  - New debugging paradigms
  - User interface scalability and design
  - Remote development environments