Developing Scientific Applications Using Eclipse and the Parallel Tools Platform

Greg Watson, IBM
g.watson@computer.org

Beth Tibbitts, IBM
tibbitts@us.ibm.com

Jay Alameda, NCSA
jalameda@ncsa.uiuc.edu

Jeff Overbey, UIUC
overbey2@illinois.edu

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, the United States Department of Energy under Contract No. DE-FG02-06ER25752, and the Blue Waters sustained petascale computing project, which is supported by the National Science Foundation under award number OCI 07-25070.
<table>
<thead>
<tr>
<th>Time (Tentative!)</th>
<th>Module</th>
<th>Outcomes</th>
<th>Presenter</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:30-8:35</td>
<td>1. Overview of Eclipse and PTP</td>
<td>✪ Introduction to Eclipse/PTP</td>
<td>Greg</td>
</tr>
<tr>
<td>8:35-8:50</td>
<td>2. Installation</td>
<td>✪ Prerequisites ✪ Installation</td>
<td>Greg</td>
</tr>
<tr>
<td>8:50-9:20</td>
<td>3. Working with C/C++</td>
<td>✪ Eclipse basics ✪ Creating a new project ✪ Building and launching</td>
<td>Beth</td>
</tr>
<tr>
<td>9:20-10:50</td>
<td>4. Working with MPI</td>
<td>✪ CVS, Makefiles, autoconf, PLDT MPI tools ✪ Resource Managers ✪ Launching a parallel application</td>
<td>Jay</td>
</tr>
<tr>
<td>10:00 - 10:30</td>
<td>Break</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:50-11:10</td>
<td>5. Fortran</td>
<td>✪ Photran overview ✪ MPI project creation ✪ Differences from CDT</td>
<td>Jeff</td>
</tr>
<tr>
<td>11:10-11:30</td>
<td>6. Debugging</td>
<td>✪ Introduction to parallel debugging ✪ Debugging an MPI program</td>
<td>Greg</td>
</tr>
<tr>
<td>11:30 – 11:50</td>
<td>7. Advanced Features</td>
<td>✪ Perspectives, Views, Preferences, Team ✪ Refactoring/Search (Fortran &amp; C/C++) ✪ PLDT (MPI, OpenMP, UPC tools) ✪ Remote Development</td>
<td>Jeff/Beth</td>
</tr>
<tr>
<td>11:50- 12:00</td>
<td>8. Other Tools, Wrapup</td>
<td>✪ NCSA HPC WB, Perf and other Tools, website, mailing lists, future features</td>
<td>Jay/Beth</td>
</tr>
</tbody>
</table>
Module 1: Introduction

- **Objective**
  - To introduce the Eclipse platform and PTP

- **Contents**
  - What is Eclipse?
  - What is PTP?
What is Eclipse?

- A vendor-neutral open-source workbench for multi-language development
- A extensible platform for tool integration
- Plug-in based framework to create, integrate and utilize software tools
Eclipse Platform

- Core frameworks and services with which all plug-in extensions are created
- Represents the common facilities required by most tool builders:
  - Workbench user interface
  - Project model for resource management
  - Portable user interface libraries (SWT and JFace)
  - Automatic resource delta management for incremental compilers and builders
  - Language-independent debug infrastructure
  - Distributed multi-user versioned resource management (CVS supported in base install)
  - Dynamic update/install service
Plug-ins

- Java Development Tools (JDT)
- Plug-in Development Environment (PDE)
- C/C++ Development Tools (CDT)
- Parallel Tools Platform (PTP)
- Fortran Development Tools (Photran)
- Test and Performance Tools Platform (TPTP)
- Business Intelligence and Reporting Tools (BIRT)
- Web Tools Platform (WTP)
- Data Tools Platform (DTP)
- Device Software Development Platform (DSDP)
- Many more...
Eclipse Parallel Tools Platform (PTP)

Coding & Analysis

Launch & Monitoring

Performance Tuning

Debugging

Module 1
Parallel Tools Platform (PTP)

- The Parallel Tools Platform aims to provide a highly integrated environment specifically designed for parallel application development.

Features include:
- An integrated development environment (IDE) that supports a wide range of parallel architectures and runtime systems
- A scalable parallel debugger
- Parallel programming tools (MPI/OpenMP)
- Support for the integration of parallel tools
- An environment that simplifies the end-user interaction with parallel systems

http://www.eclipse.org/ptp
Module 2: Installation

- **Objective**
  - To learn how to install Eclipse and PTP

- **Contents**
  - System Prerequisites
  - Software Prerequisites
  - Eclipse Installation
  - PTP Installation
About PTP Installation

- PTP 3.0 isn’t “official” yet. Planned for late Oct.

- Note: up-to-date info on installing PTP and its pre-reqs is available from the release notes:
  
  http://wiki.eclipse.org/PTP/release_notes/3.0

- This information may supersede these slides
System Prerequisites

- Local system (running Eclipse)
  - Linux (just about any version)
  - MacOSX (Leopard)
  - Windows (XP on)

- Remote system (running/debugging application)
  - Must be supported by a resource manager
  - Open MPI 1.2+
  - MPICH 2
  - IBM PE & LoadLeveler (AIX or Linux)
  - SLURM (Linux)
Software Prerequisites

- Java (1.5 or later)
- Cygwin or MinGW (for local development on Windows)
- Unix make or equivalent
- Supported compilers (gcc, gfortran, Intel, etc.)
- Gdb for debugging (or a gdb-like interface)
- Gcc for building the debugger and SLURM proxies from source
- IBM C for building the PE/LoadLeveler proxies from source
Java Prerequisite

- Eclipse requires Sun or IBM versions of Java
  - Only need Java runtime environment (JRE)
  - Java 1.5 is the same as JRE 5.0
  - The GNU Java Compiler (GCJ), which comes standard on Linux, will not work!
Eclipse and PTP Installation

- Eclipse is installed in two steps
  - First, the base Eclipse package is downloaded and installed
  - Additional functionality is obtained by adding ‘features’
    - This can be done via an ‘update site’ that automatically downloads and installs the features
    - Update site archives can be downloaded to install features offline.

- PTP requires the following Eclipse features
  - C/C++ Development Tools (CDT)
  - Remote Systems Explorer (RSE) end-user runtime
Eclipse Packages

- Eclipse is available in a number of different packages for different kinds of development
- Two packages are more relevant for HPC:
  - **Eclipse Classic**
    - The full software development kit (SDK), including Java and Plug-in development tools
  - **Eclipse IDE for C/C++ developers**
    - Base Eclipse distribution
    - Base C/C++ Development Tools (CDT) (does not include UPC)
- Either is ideal for PTP use
Eclipse Installation

- The current version of Eclipse is 3.5 (Galileo)
  - PTP 3.0 will only work with this version
- Eclipse is downloaded as a single zip or gzipped tar file from [http://eclipse.org/downloads](http://eclipse.org/downloads)
- You must download the correct version to suit your local environment
  - Must have correct operating system version
  - Must have correct window system version
- Unzipping or untarring this file creates a directory containing the main executable
Platform Differences

- Single button mouse (e.g. MacBook)
  - Use Control-click for right mouse / context menu
- Context-sensitive help key differences
  - Windows: use F1 key
  - Linux: use Shift-F1 keys
  - MacOS X
    - Full keyboard, use Help key
    - MacBooks or aluminum keyboard, create a key binding for Dynamic Help to any key you want
- Accessing preferences
  - Windows & Linux: Window ► Preferences…
  - MacOS X: Eclipse ► Preferences…
Starting Eclipse

- **Linux**
  - From a terminal window, enter
  
  ```
  <eclipse_installation>/eclipse/eclipse &
  ```

- **MacOS X**
  - From finder, open the `eclipse` folder where you installed
  - Double-click on the `Eclipse` application
  - Or from a terminal window

- **Windows**
  - Open the `eclipse` folder
  - Double-click on the `eclipse` executable

- Accept default workspace when asked
- Select workbench icon from welcome page
Specifying A Workspace

- Eclipse prompts for a workspace location at startup time
- The workspace contains all user-defined data
  - Projects and resources such as folders and files

The prompt can be turned off.
Eclipse Welcome Page

(Displayed when Eclipse is run for the first time)

Select “Go to the workbench”
Adding Features

- New functionality is added to Eclipse using *features*
- Features are obtained and installed from an update site (like a web site)
- Features can also be installed from a local copy of the update site (which can be zipped)
Installing Eclipse Features from an Update Site

- Three types of update sites
  - **Remote** - download and install from remote server
  - **Local** - install from local directory
  - **Archived** - a local site packaged as a zip or jar file

- Eclipse 3.5 comes preconfigured with a link to the **Galileo** Update Site
  - This is a remote site that contains a large number of official features
  - Galileo projects are guaranteed to work with Eclipse 3.5

- Many other sites offer Eclipse features
  - Use at own risk
Installing from an Update Site

From the **Help** menu, choose **Install New Software...**
Galileo Update Site

- The Galileo site comes already configured with Eclipse
- For example, some of the contents of the Galileo site:
  - You can get C/C++ Dev. Tools from the Galileo site, but...
    - Basic tools, does not include UPC
    - More complete CDT install shown later
    - PTP 3.0 needs CDT 6.0.1, not available yet

Module 2
Installation: RSE

The RSE End-User Runtime should be installed from the Galileo update site.
Installation: CDT

- PTP 3.0 needs CDT 6.0.1
  - Update site will contain latest version
- Update site: http://download.eclipse.org/tools/cdt/releases/galileo
- Install any features you want
  - Omit the testing feature:
    - Eclipse CDT Testing Feature
- If you want UPC, include:
  - Unified Parallel C Support
- CDT 6.0.1 is due at the end of Sept
Installing PTP

- In **Work with** type the PTP update site URL: http://download.eclipse.org/tools/ptp/releases/galileo/
- Click **Add...**
- Enter a name (optional) e.g. “PTP 3.0”
- Click **OK** and the list of features on the update site will be populated
- Select all the components you require
- Click **Next>**

- See PTP release notes for most recent info on installing 3.0
Installing PTP (2)

- You will be prompted to accept the License terms
- Accept the License terms
- Click **Finish**

- Restart Eclipse when prompted
Restarting Eclipse

- Welcome page informs you of new features installed
- Select workbench icon to go to workbench

Go to workbench

Yellow indicates new features just installed
Installing Additional PTP Components

- PTP has a number of additional components depending on the installation
  - Scalable Debug Manager (SDM) – required for all platforms to support debugging
  - PE and LoadLeveler proxy – IBM systems only
  - SLURM proxy – systems using the SLURM resource manager
- Installation of these components is beyond the scope of the tutorial
- See the release notes for details of installing these components
Module 3: Working with C/C++

Objective
- Learn how to use Eclipse to develop parallel programs
- Learn how to run and monitor a parallel program

Contents
- Brief introduction to the C/C++ Development Tools
- Create a simple application
- Learn to launch a parallel job and view it via the PTP Runtime Perspective
Installation recap

- Download and unzip/untar eclipse
- Use Help > Install new software to get
  - CDT for C/C++ tools
  - PTP and related tools for Parallel application work
  - Build PTP binary on target machine (local or remote)
- Launch eclipse!
  Run the ‘eclipse’ executable, from icon or from command line
The Workbench represents the desktop development environment:
- It contains a set of tools for resource management.
- It provides a common way of navigating through the resources.
- Multiple workbenches can be opened at the same time.
Workbench Components

- A Workbench contains perspectives
- A Perspective contains views and editors
Perspectives define the layout of views in the Workbench. They are task oriented, i.e., they contain specific views for doing certain tasks:
- There is a Resource Perspective for manipulating resources.
- C/C++ Perspective for manipulating compiled code.
- Debug Perspective for debugging applications.
You can easily switch between perspectives.
Switching Perspectives

- You can switch Perspectives by:
  - Choosing the **Window ➤ Open Perspective** menu option
  - Clicking on the **Open Perspective** button
  - Clicking on a perspective shortcut button
Available Perspectives

- By default, certain perspectives are available in the Workbench
- We’ll use:
  - C/C++
  - PTP Runtime
  - PTP Debug
Views

- The workbench window is divided up into Views
- The main purpose of a view is:
  - To provide alternative ways of presenting information
  - For navigation
  - For editing and modifying information
- Views can have their own menus and toolbars
  - Items available in menus and toolbars are available only in that view
  - Menu actions only apply to the view
- Views can be resized
Stacked Views

- Stacked views appear as tabs
- Selecting a tab brings that view to the foreground
Help

- Access help
  - Help ▶ Help Contents
  - Help ▶ Search
  - Help ▶ Dynamic Help
- Help Contents provides detailed help on different Eclipse features
- Search allows you to search for help locally, or using Google or the Eclipse web site
- Dynamic Help shows help related to the current context (perspective, view, etc.)

Module 3
Switch to C/C++ Perspective

- Only needed if you’re not already in the perspective

- What Perspective am I in? See Title Bar
Project Explorer View

- Represents user’s data
- It is a set of user defined resources
  - Files
  - Folders
  - Projects
    - Collections of files and folders
    - Plus meta-data
- Resources are visible in the Project Explorer View
Editors

✧ An editor for a resource (e.g. a file) opens when you double-click on a resource
✧ The type of editor depends on the type of the resource
  ✧ .c files are opened with the C/C++ editor
  ✧ Some editors do not just edit raw text
✧ When an editor opens on a resource, it stays open across different perspectives
✧ An active editor contains menus and toolbars specific to that editor
✧ When you change a resource, an asterisk on the editor’s title bar indicates unsaved changes

Module 3
Source Code Editors

- A source code editor is a special type of editor for manipulating source code
- Language features are highlighted
- Marker bars for showing
  - Breakpoints
  - Errors/warnings
  - Tasks
- Location bar for navigating to interesting features

Icons:
- Task tag
- Warning
- Error

Module 3
Preferences

- Eclipse Preferences allow customization of almost everything
  - Open Window ▶ Preferences...
  - C/C++ preferences allow many options
  - Code formatting settings (“Code Style”) shown here

Module 3
Creating a C/C++ Application

Steps:
- Create a new C project
- Edit source code
- Save and build
New C Project Wizard

Create a new MPI project

- **File ➤ New ➤ C Project** (see prev. slide)
- Name the project ‘MyHelloProject’
- Under Project types, under Executable, select **Hello World ANSI C Project** (no makefile req’d) and hit **Next**
- On **Basic Settings** page, fill in information for your new project (**Author name** etc.) and hit **Next**

This is a “Managed Build” project

Makefile project; provide your own makefile
Changing the C/C++ Build Settings Manually

- Open the project properties by right-mouse clicking on project and select **Properties**
- Open **C/C++ Build**
- Select **Settings**
- Select **C Compiler** to change compiler settings
- Select **C Linker** to change linker settings
- It’s also possible to change compiler/linker arguments
- Hit **OK** to close

Module 3
Editor and Outline View

- Double-click on source file in the **Project Explorer** to open C editor
- Outline view is shown for file in editor
Content Assist

- Type an incomplete function name e.g. "get" into the editor, and hit **ctrl-space**
- Select desired completion value with cursor or mouse

- Hover over a program element in the source file to see additional information
Create a Launch Configuration

- Open the run configuration dialog Run ➤ Run Configurations...
- Select C/C++ Application
- Select the New button

Depending on which flavor of Eclipse you installed, you might have more choices in Application types.
Complete the Main Tab

- Ensure that the correct project is selected.
- Select the **C/C++ Application** (executable) if necessary.
  - **Search Project...** will search just within the project.
  - **Browse** will search anywhere on the local file system.
- Select **Connect process input/output to a terminal** if desired.
Complete the Arguments Tab

- Enter any program arguments into the text box.
- Eclipse variables can also be passed using the **Variables...** button.
- Select a different working directory if desired.
Complete the Debugger Tab

- Select **Debugger** tab
- Make sure **gdb/mi** is selected
- Change where the program should stop if desired
- Change any gdb-specific options if desired (advanced users only)

The information on the debugger tab will only be used for a debug launch

- Hit the Run button to launch your program

宗 Module 3
Viewing Program Output

- When the program runs, the **Console** view should automatically become active
- Any output will be displayed in this view (stderr in red)
Module 4: Working with MPI

**Objective**
- Learn how to build and launch an MPI program
- Explore some of the features to aid MPI programming

**Contents**
- Using a version control system (CVS)
- Building with Makefiles and autoconf
- MPI assistance features
- Working with resource managers
- Launching a parallel application
Creating the Project

- Configuring version control
- Checking out the source code
- Team support
Connecting to a Repository

- Select **Window ➤ Open Perspective ➤ Other...**
- Select **CVS Repository Exploring** then **OK**
Specify Repository Location

- Right-click in the **CVS Repositories** view, then select **New ➤ Repository Location**...
- Set **Host** to the hostname of remote machine
- Set **Repository path** to the CVS repository path
- Fill in **Username** and **Password**
- Set **Connection type** to **extssh** to use an ssh connection
- Check **Save password** if you wish to save the password
- Select **Finish**
CVS Repository Exploring

- Open the repository in the **CVS Repository** view
- Open **HEAD** to view files and folders in the CVS head
- Open **Branches** or **Versions** to view CVS branches or versions respectively
- Right-click on the repository and select **Refresh Branches...** to see all branches and versions

*Module 4*
Check out as an Eclipse Project

- In CVS Repositories view, right-click on project and select **Project ➤ Check out As**...
- Make sure **Check out as a project configured using the New Project Wizard** is selected
- Leave **Checkout subfolders** checked
- Select **Finish**
Create a C Project

- The **New Project Wizard** is used to create a C project.

- Enter **Project name**
- Under **Project Types**, select **Makefile project > Empty Project**
  - Ensures that CDT will use existing makefiles
- Select **Finish**
- When prompted to switch to the **C/C++ Perspective**, select **Yes**
Building the Application

- Configuring the project build directory
- Generating Makefiles
- Creating a Make Target
- Running the build
Create a **build** directory

- This program requires a separate build directory
- Select the project in the **Project Explorer** view
- From the **File** menu, select elect **New ➤ Folder**…
- Make sure the parent folder is correct
- Enter “build” as the folder name
- Click **Finish**
Makefile Project

- Similar to managed project, but uses custom Makefile (or other script) to control build
- User can specify command that will be used to initiate build
- Can also specify the directory in which the build will take place
- “Make targets” are used to control type of build
- Can switch between managed and unmanaged project
Makefile Project Properties

- Right click on project in Project Explorer to bring up properties
- Click on C/C++ Build for the build settings
- Can change build command if desired
- Change the Build location to the build directory in the project
About Makefiles and autoconf

- **Autoconf** is a GNU utility often used to create Makefiles for open source projects
  - Used to generate a `configure` script
  - `Configure` is run to generate a Makefile that suits a particular system configuration
  - Normally only needs to be run once, unless the build process needs to be changed
- **Run `configure` using two methods:**
  - Manually from an external shell
  - By creating an **External Tools Launch Configuration**
- **Must refresh** **Project Explorer** whenever file system is modified outside of Eclipse, such as after running `configure`
Generate the Makefiles

- From the Run menu, select External Tools ➤ External Tools Configurations…
- Create a new Program
- For Location, click Browse Workspace… and find the configure script
- For Working Directory, click Browse Workspace… and select the build directory in the project
- Click Run and you should see output in the Console view
- In Project Explorer, right-click and select Refresh to see the new files that have been created
Create a Make Target

- Select the project in **Make Targets** view
- Click on **New Make Target** icon
- Enter the desired name of the target
- Unselect Same as the target name and delete “build”
  - This will run the “make” command with no arguments
- Select **OK**
Running the Build

- Open the project in the **Make Targets** view to see the **build** target
- Double-click on the **build** target to initiate the build
- Output from the build will be visible in the **Console** view
MPI Assistance Tools

Added by PLDT (Parallel Lang. Dev. Tools) feature of PTP

- MPI Context sensitive help
- MPI artifact locations
- MPI barrier analysis
- MPI templates
Context Sensitive Help

- Click mouse, then press help key when the cursor is within a function name
  - Windows: F1 key
  - Linux: ctrl-F1 key
  - MacOS X: Help key or Help ➤ Dynamic Help

- A help view appears (Related Topics) which shows additional information (You may need to click on MPI API in editor again, to populate)

- Click on the function name to see more information

- Move the help view within your Eclipse workbench, if you like, by dragging its title tab

Module 4

Some special info has been added for MPI APIs.
Show MPI Artifacts

- Select source file; Run analysis by clicking on drop-down menu next to the analysis button and selecting **Show MPI Artifacts**
- Markers indicate the location of artifacts in editor
- In **MPI Artifact View** sort by any column (click on col. heading)
- Navigate to source code line by double-clicking on the artifact
- Run the analysis on another file and its markers will be added to the view
- Remove markers via **Module 4**
MPI Barrier Analysis

Verify barrier synchronization in C/MPI programs

Interprocedural static analysis outputs:

- For verified programs, lists barrier statements that synchronize together (match)
- For synchronization errors, reports counter example that illustrates and explains the error
MPI Barrier Analysis - views

MPI Barriers view
Simply lists the barriers
Like MPI Artifacts view, double-click to navigate to source code line (all 3 views)

Barrier Matches view
Groups barriers that match together in a barrier set – all processes must go through a barrier in the set to prevent a deadlock

Barrier Errors view
If there are errors, a counter-example shows paths with mismatched number of barriers

Module 4
MPI Templates

- Allows quick entry of common patterns in MPI programming
- Example: MPI send-receive
- Enter: mpisr <ctrl-space>
- Expands to

```c
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &p);
if (rank == 0){ //master task
    printf("Hello From process 0: Num processes: %d\n",p);
    for (source = 1; source < p; source++) {
        MPI_Recv(message, 100, MPI_CHAR, source, tag,
                  MPI_COMM_WORLD, &status);
        printf("%s\n",message);
    }
    else{ // worker tasks
        /* create message */
        sprintf(message, "Hello from process %d!", my_rank);
        dest = 0;
        /* use strlen+1 so that '\0' get transmitted */
        MPI_Send(message, strlen(message)+1, MPI_CHAR,
                  dest, tag, MPI_COMM_WORLD);
    }
}
```

- Eclipse preferences: add more!
  - C/C++ > Editor > Templates
  - Extend to other common patterns

Module 4
Running the Program

- Terminology
- PTP Runtime Perspective
- Resource Managers
- Launch Configurations
Terminology

- The **PTP Runtime** perspective is provided for monitoring and controlling applications.

Some terminology:
- **Resource manager** - Corresponds to an instance of a resource management system (e.g., a job scheduler). You can have multiple resource managers connected to different machines.
- **Queue** - A queue of pending jobs.
- **Job** - A single run of a parallel application.
- **Machine** - A parallel computer system.
- **Node** - Some form of computational resource.
- **Process** - An execution unit (may be multiple threads of execution).
Resource Managers

- PTP uses the term “resource manager” to refer to any subsystem that controls the resources required for launching a parallel job.
- Examples:
  - Job scheduler (e.g. LoadLeveler)
  - Open MPI Runtime Environment (ORTE)
- Each resource manager controls one target system
- Resource Managers can be local or remote
About PTP Icons

Open using legend icon in toolbar
Open PTP Runtime Perspective

Window > Open Perspective > Other...

Module 4
PTP Runtime Perspective

- Resource managers view
- Machines view
- Node details view
- Jobs view
Adding a Resource Manager

- Right-click in Resource Managers view and select Add Resource Manager

- Choose the Open MPI Resource Manager Type

- Select Next >
Configure the Remote Location

- Choose **RSE** for *Remote service provider*
- Choose **Remote location** or click **New...** to create a new location
  - **Local** can be used to run applications locally
- Some resource managers support tunneling over ssh connections (e.g. Remote Tools)
- The port forwarding option would be enabled if it was available
Create a New Location (RSE)

- Choose **SSH Only** for this connection
- Click **Next>**
- Enter **Host name** of remote system
- Click **Finish**

Module 4
The Open MPI resource manager will auto detect the version and use the appropriate commands.

- Change only if you’re an expert
- Click Next>
- Change the **Name** or **Description** of the resource manager if you wish
- Click Finish
Starting the Resource Manager

- Right click on new resource manager and select **Start resource manager**
- If everything is ok, you should see the resource manager change to **green**
- If something goes wrong, it will change to **red**
System Monitoring

- Machine status shown in **Machines** view
- Node status also shown in **Machines** view
- Hover over node to see node name
- Double-click on node to show attributes
Open the run configuration dialog Run ➤ Run Configurations...
- Select Parallel Application
- Select the New button

Depending on which flavor of Eclipse you installed, you might have more choices in Application types.
Complete the Resources Tab

- In **Resources** tab, select the resource manager you want to use to launch this job
- Enter a value in the **Number of processes** field
- Other fields can be used to specify resource manager-specific information
Complete the Application Tab

- Select the **Application** tab
- Choose the **Application program** (executable) by clicking the **Browse** button
  - Local program: executable is under Debug folder in the project
  - Remote program: must copy to remote machine; navigate to its location on the remote machine here
- Select **Display combined output in a console view** if desired
Complete the Debugger Tab

- Select **Debugger** tab
- Choose **SDM** from the **Debugger** dropdown
- Use the **Browse** button to select the debugger executable
  - If launching remotely, the debugger executable must also be located remotely
- Set debugger session address (covered later)
- Click on **Run** to launch the program

The debugger settings will not be used until the application is launched under the debugger
This will be covered in more detail in Module 6
Viewing The Run

- Double-click a node in machines view to see which processes ran on the node
- Hover over a process for tooltip popup
- Job and processes shown in jobs view
Viewing Program Output

- Double-click a process to see process detail and standard output from the process.
- Console displays combined output from all processes.
Module 5: Fortran

- **Objective**
  - Learn what Photran is and how it compares to CDT
  - Learn how to create a Fortran MPI application

- **Contents**
  - Overview of Photran
  - Module 3 redux (in Fortran)
  - Differences between Photran and CDT
  - Pointers to online documentation for Photran
Photran

- [http://www.eclipse.org/photran](http://www.eclipse.org/photran)
- Official Eclipse Foundation project; part of the Parallel Tools Platform (PTP)
- 20,000 downloads/release (2007)
- Supports Fortran 77, 90, 95, and 2003
- Built on CDT; largely similar to it
- Primary contributor: UIUC
- Contrib’s from Intel, IBM, LANL, & others
Module 5

Fortran Editor & Outline
Module 5

### Context-Aware Highlighting

A code snippet showing fixed format source with context-aware highlighting:

```fortran
c Fixed format source with context-aware highlighting
integer :: if = 3
integer :: end = 5
character :: endif = "Hello"
if (if .gt. end) then
  print *, &
 endif
endif
end
```

- **Fixed Form Support**
- **Context-Aware Highlighting**
CVS support
Module 5

Debugging (GDB GUI)
Using Photran

- It’s just like using CDT...
  - Similar New Project wizards
  - Similar build procedure
  - Similar launch/debug procedure

- ...but not exactly
  - Configuring fixed vs. free form file extensions
  - Different editor features
  - Different advanced features (Module 7)
Switch to C/C++ Perspective

- Only needed if you’re not already in the perspective

- What Perspective am I in? See Title Bar
Creating a C/C++ Application

Steps:

- Create a new C project
- Edit source code
- Save and build
Module 5

Create a new MPI project

- **File ➤ New ➤ C Project** (see prev. slide)
- Name the project ‘MyHelloProject’
- Under Project types, under Executable, select MPI Hello World C Project
- On **Basic Settings** page, fill in information for your new project (**Author name** etc.) and hit **Next**

There are “Managed Build” projects for Fortran too…

…but this is a Makefile project, where you maintain the Makefile
Module 5

Fortran Projects

Project Explorer View

- Represents user’s data
- It is a set of user defined resources
  - Files
  - Folders
  - Projects
    - Collections of files and folders
    - Plus meta-data
- Resources are visible in the Project Explorer View

Fortran Projects
Editor and Outline View

- Double-click on source file to open editor
- Outline view is shown for file in editor
Et Cetera

Building (compiling) is identical

**Tip:** Are compile errors not shown in the Problems view?

- Right-click on the project in the Fortran Projects view, and choose **Properties**
- Expand **Fortran Build** ➤ **Settings**
- Switch to the **Error Parsers** tab
- Are Photran’s error parsers checked? If not, click **Check all**
- Click **OK** and re-build
Creating a launch configuration is identical

(Suggestion: Uncheck **Stop on startup at main** in the Debugger tab)

**Tip:** Is your binary not listed when you create a launch configuration?

- Right-click on the project in the Fortran Projects view, and choose **Properties**
- Expand **Fortran Build** ➤ **Settings**
- Switch to the **Binary Parsers** tab
- Make sure the parser for your platform is checked
  - PE = Windows
  - Elf = Linux
  - Mach-O = Mac OS X
- Click **OK**
Et Cetera

- Debugging is identical
- Launching a parallel application is identical
- Debugging a parallel debugging is identical
Differences (1): MPI Project Wizard

- In the MPI Hello World C Project, the MPI compiler is set in the project settings...  
  (See “Changing the C/C++ Build Settings Manually” in Module 3)

- ...but in the MPI Hello World Fortran Project, the MPI compiler is set in a Makefile.
Differences (2): Content Assist

- Content assist is *disabled* by default.
  (So are Declaration View, Hover Tips, Fortran Search, and refactoring.)
You must specifically enable it for your project.

- Right-click on the project in the Fortran Projects view, and choose **Properties**
- Expand **Fortran Analysis/Refactoring**
- Check **Enable Fortran analysis/refactoring**
- Click **OK**
- Close and re-open any Fortran editors
Differences (3): Source Form

- Fortran files are either *free form* or *fixed form*
  - Determined by filename extension
  - Extensions are set in the workspace preferences

- Defaults:
  - Fixed form: .f .fix .for .fpp .ftn
  - Free form: .f03 .f95 .f90 .f77

- Many features *will not work* if filename extensions are associated incorrectly
  (Outline view, content assist, Fortran Search, refactorings, Open Declaration, ...)

*Module 5*
Differences (3): Source Form

Set fixed/free form filename extensions in the preferences

- Navigate the tree to General ▶ Content Types
- Expand Text ▶ Fortran Source File
- Select Fixed or Free Format and set associations

Module 5
Differences (3): Source Form

Outline view displays expected source form of file in editor
(according to the workspace preferences)

Attempting to open a file with the “wrong” editor
(right-click ➤ **Open With**) issues a warning.
For More Information

- **Module 7**: Fortran Search, Refactoring

- **Photran online documentation**
  linked from http://www.eclipse.org/photran

  - **User’s Guide**
    General introduction, basic features

  - **Advanced Features Guide**
    Features requiring analysis/refactoring to be enabled

- **Online tutorial**: Compiling and running the Parallel Ocean Program using Photran and PTP
  linked from http://wiki.eclipse.org/PTP/photran/tutorials
Module 6: Parallel Debugging

- **Objective**
  - Learn the basics of debugging parallel programs with PTP

- **Contents**
  - Launching a parallel debug session
  - The PTP Debug Perspective
  - Controlling sets of processes
  - Controlling individual processes
  - Parallel Breakpoints
  - Terminating processes
Launching A Debug Session

- Use the drop-down next to the debug button (bug icon) instead of run button
- Select the project to launch
- The debug launch will use the same number of processes that the normal launch used (edit the [Debug Launch Configuration](#) to change)
The PTP Debug Perspective (1)

- **Parallel Debug view** shows job and processes being debugged
- **Debug view** shows threads and call stack for individual processes
- **Source view** shows a **current line marker** for all processes
The PTP Debug Perspective (2)

- **Breakpoints** view shows breakpoints that have been set (more on this later).
- **Variables** view shows the current values of variables for the currently selected process in the **Debug** view.
- **Outline** view (from CDT) of source code.
Stepping All Processes

- The buttons in the **Parallel Debug View** control groups of processes
- Click on the **Step Over** button
- Observe that all process icons change to green, then back to yellow
- Notice that the current line marker has moved to the next source line
Stepping An Individual Process

- The buttons in the **Debug view** are used to control an individual process, in this case process 0
- Click the **Step Over** button
- You will now see two current line markers, the first shows the position of process 0, the second shows the positions of processes 1-3
Traditional debuggers apply operations to a single process.

Parallel debugging operations apply to a single process or to arbitrary collections of processes.

A process set is a means of simultaneously referring to one or more processes.
When a parallel debug session is first started, all processes are placed in a set, called the **Root** set.

- Sets are always associated with a single job.
- A job can have any number of process sets.
- A set can contain from 1 to the number of processes in a job.
Operations On Process Sets

- Debug operations on the **Parallel Debug view** toolbar always apply to the current set:
  - Resume, suspend, stop, step into, step over, step return
- The current process set is listed next to job name along with number of processes in the set
- The processes in process set are visible in right hand part of the view

Root set = all processes
Managing Process Sets

The remaining icons in the toolbar of the **Parallel Debug view** allow you to create, modify, and delete process sets, and to change the current process set.

- Create set
- Remove from set
- Change current set
- Delete set
Creating A New Process Set

- Select the processes you want in the set by clicking and dragging, in this case, the last three.
- Click on the **Create Set** button.
- Enter a name for the set, in this case **workers**, and click **OK**.
- You will see the view change to display only the selected processes.
Stepping Using New Process Set

- With the **workers** set active, click the **Step Over** button.
- You will see only the first current line marker move.
- Step a couple more times.
- You should see two line markers, one for the single master process, and one for the 3 worker processes.
Process Registration

- Process set commands apply to groups of processes.
- For finer control and more detailed information, a process can be registered and isolated in the **Debug view**.
- Registered processes, including their stack traces and threads, appear in the **Debug view**.
- Any number of processes can be registered, and processes can be registered or un-registered at any time.
Registering A Process

- To register a process, double-click its process icon in the **Parallel Debug view** or select a number of processes and click on the **register** button.
- The process icon will be surrounded by a box and the process appears in the **Debug view**.
- To un-register a process, double-click on the process icon or select a number of processes and click on the **unregister** button.

*Module 6*
Current Line Marker

- The current line marker is used to show the current location of suspended processes
- In traditional programs, there is a single current line marker (the exception to this is multi-threaded programs)
- In parallel programs, there is a current line marker for every process
- The PTP debugger shows one current line marker for every group of processes at the same location
Colors And Markers

- The highlight color depends on the processes suspended at that line:
  - **Blue**: All registered process(es)
  - **Orange**: All unregistered process(es)
  - **Green**: Registered or unregistered process with no source line (e.g. suspended in a library routine)

- The marker depends on the type of process stopped at that location

- Hover over marker for more details about the processes suspend at that location
Breakpoints

- Apply only to processes in the particular set that is active in the **Parallel Debug view** when the breakpoint is created.
- Breakpoints are colored depending on the active process set and the set the breakpoint applies to:
  - **Green** indicates the breakpoint set is the same as the active set.
  - **Blue** indicates some processes in the breakpoint set are also in the active set (i.e. the process sets overlap)
  - **Yellow** indicates the breakpoint set is different from the active set (i.e. the process sets are disjoint)
- When the job completes, the breakpoints are automatically removed.
Creating A Breakpoint

- Select the process set that the breakpoint should apply to, in this case, the **workers** set.
- Double-click on the left edge of an editor window, at the line on which you want to set the breakpoint, or right click and use the **Parallel Breakpoint Toggle Breakpoint** context menu.
- The breakpoint is displayed on the marker bar.
Hitting the Breakpoint

- Click on the **Resume** button in the **Parallel Debug view**
- In this example, the three worker processes have hit the breakpoint, as indicated by the yellow process icons and the current line marker
- Process 0 is still running as its icon is green
- Processes 1-3 are suspended on the breakpoint
The **Step** buttons are only enabled when all processes in the active set are **suspended** (yellow icon).

In this case, process 0 is still running.

Switch to the set of suspended processes (the **workers** set).

You will now see the **Step** buttons become enabled.
Breakpoint Information

- Hover over breakpoint icon
  - Will show the sets this breakpoint applies to
- Select **Breakpoints** view
  - Will show all breakpoints in all projects
Breakpoints View

- Use the menu in the breakpoints view to group breakpoints by type
- Breakpoints sorted by breakpoint set (process set)
Global Breakpoints

- Apply to all processes and all jobs
- Used for gaining control at debugger startup
- To create a global breakpoint
  - First make sure that no jobs are selected (click in white part of jobs view if necessary)
  - Double-click on the left edge of an editor window
  - Note that if a job is selected, the breakpoint will apply to the current set

```c
if (my_rank != 0) {
    /* create message */
    sprintf(message, "Greeting ");
}
```
Terminating A Debug Session

- Click on the **Terminate** icon in the **Parallel Debug view** to terminate all processes in the active set.
- Make sure the **Root** set is active if you want to terminate all processes.

- You can also use the **terminate** icon in the **Debug view** to terminate the currently selected process.
Module 7: Advanced Development

Objective
- Explore some of the advanced features of Eclipse and PTP

Contents
- Advanced Eclipse Concepts (generic, not CDT/PTP)
- Refactoring and Search in Fortran and C/C++
- Parallel Language Development Tools: MPI, OpenMP, UPC
- Special Tools for parallel development
- Remote Development
Advanced Eclipse Concepts

- Perspectives, views and customizing
- Workbench Preferences
- Version Control
- Task Tags
Customizing Perspectives

- Items such as shortcuts, menu items and views may be customized
  - **Window»Customize Perspective...**
- Save changes
  - **Window»Save Perspective As...**
- Close Perspective
  - Right-click on perspective title and select **Close**
- Reset Perspective
  - **Window»Reset Perspective** resets the current perspective to its default layout
Opening New Views

To open a view:
- Choose **Window ► Show View ► Other...**
- The **Show View** dialog comes up
- Select the view to be shown
- Select **OK**
Workbench Preferences

- Preferences provide a way for you to customize your Workbench
  - By selecting **Window ➤ Preferences**... or **Eclipse ➤ Preferences**... (Mac)
- Examples of preference settings
  - Use Emacs bindings for editor **keys**
  - Modify editor folding defaults
    - E.g., fold all macro definitions
  - Associate file types with file extensions
    - E.g., *.f03 with the Fortran editor
  - Toggle automatic builds
  - Change key sequence shortcuts
    - E.g., Ctrl+/ for Comment
Version Control
(Eclipse Team Support)

- Version control provided through the **Project Explorer** view, in the **Team** context menu
- Provides familiar actions:
  - Commit...
  - Update...
- Also less used tasks:
  - Create/Apply Patch...
  - Tag as Version...
  - Branch...
  - Merge...
  - Add to .cvsignore...

Module 4
Team Synchronizing

- Accessed from the Team ▶ Synchronize with Repository context menu
- Shows:
  - Files to be added
  - Files to be updated
  - Files to be committed
  - Files to be deleted
  - Merge conflicts
- Double-click on file to show compare viewer
- Operations can be performed on individual files or all at once

Module 4
Task Tags

- Task tags are identifiers in C/C++ comments
- TODO is a built-in task tag
- The build locates task tags during compilation
- View task tags in Tasks View
  - If it’s not shown, Window ➤ Show View ➤ Other... Open General and select Tasks
- Configure your own task tag in Window ➤ Preferences
  - Under C/C++, select Task Tags

Module 7
Refactoring and Search in Fortran and C/C++
Refactoring
(making changes to source code that don’t affect the behavior of the program)

- Refactoring is the research motivation for Photran @ Illinois
  - Illinois is a leader in refactoring research
  - “Refactoring” was coined in our group (Opdyke & Johnson, 1990)
  - We had the first dissertation... (Opdyke, 1992)
  - ...and built the first refactoring tool... (Roberts, Brant, & Johnson, 1997)
  - ...and first supported the C preprocessor (Garrido, 2005)
  - Photran’s agenda: refactorings for HPC, language evolution, refactoring framework

- Photran 5.0: 10-15 refactorings
Rename Refactoring

- Changes the name of a variable, function, etc., including every use.
  (change is semantic, not textual, and can be workspace-wide)

- Only proceeds if the new name will be legal.
  (aware of scoping rules, namespaces, etc.)

Select C/C++ Perspective or Fortran Perspective
Open a source file
Click in editor view on declaration of a variable
Select menu item Refactor ▶ Rename
Or use context menu
Enter new name
Extract Procedure Refactoring

- Moves statements into a new subroutine, replacing the statements with a call to that subroutine
- Local variables are passed as arguments

Select a sequence of statements
Select menu item
**Refactor** ➤ **Extract Procedure...**
in Fortran, or, in C/C++,
**Refactor** ➤ **Extract Function...**
- Or use context menu
- Enter new name
Photran Implicit Refactoring

- Introduce Implicit None adds an IMPLICIT NONE statement and adds explicit variable declarations for all implicitly declared variables.

- Introduce in a single file by opening the file and selecting Refactor ➤ Introduce IMPLICIT NONE...

- Introduce in multiple files by selecting them in the Fortran Projects view, right-clicking on the selection, and choosing Refactor ➤ Introduce IMPLICIT NONE...
CDT Rename in File

- Position the caret over an identifier.
- Press Ctrl+1 (Command+1 on Mac).
- Enter a new name. Changes are propagated within the file as you type.
CDT Extract Constant Refactoring

The following changes are necessary to perform the refactoring.

Changes to be performed

- Changes
- MyCproject.c - MyCproject/src

Original Source

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    double intvalue = 0.0;
    puts("Hello World!"); /* print ! */
    return EXIT_SUCCESS;
}
int foo() {
    double myint = 0.0;
}
```

Refactored Source

```c
#include <stdio.h>
#include <stdlib.h>

static const float MYZERO = 0.0;

int main(void) {
    double intvalue = MYZERO;
    puts("Hello World!"); /* print ! */
    return EXIT_SUCCESS;
}
int foo() {
    double myint = MYZERO;
}
```

Other refactorings that are planned:

- Extract Function
- Hide Member Function
- Move Field or Member Function
- Extract Subclass
- Extract Baseclass
- Separate Class
- Implement Function
- Declare Function
- Move Function Definition
- Generate Getters and Setters
“Knows” what things can be declared in each language (functions, variables, classes, modules, etc.)

- E.g., search for every call to a function whose name starts with “get”
- Search can be project- or workspace-wide

Module 7

PTP Tutorial
Open Declaration

- Jumps to the declaration of a variable, function, etc., even if it’s in a different file
- Right-click on an identifier
- Click Open Declaration

Tip: Open Declaration works in C/C++, and it works in Fortran, but it cannot jump “across languages”
Parallel Lang. Dev. Tools

- **PLDT Features**
  - Analysis of C and C++ code to determine the location of MPI, OpenMP, and UPC Artifacts
  - Content assist via `ctrl+space` ("completion")
  - Hover help
  - Reference information about the API calls via Dynamic Help
  - New project wizard automatically configures managed build projects for MPI & OpenMP
  - OpenMP problems view of common errors
  - OpenMP "show #pragma region", "show concurrency"
  - MPI Barrier analysis - detects potential deadlocks

Some MPI features were covered in Module 4
OpenMP Managed Build Project

- If you haven’t set up OpenMP preferences e.g. include file location, do it now
- Create a new OpenMP project
  - File ➔ New ➔ C Project
  - Name the project e.g. ‘MyOpenMPproject’
  - Select OpenMP Hello World C Project
  - Select Next, then fill in other info like MPI project
Setting OpenMP Special Build Options

- OpenMP typically requires special compiler options.
  - Open the project properties
  - Select **C/C++ Build**
  - Select **Settings**
  - Select **C Compiler**
  - In Miscellaneous, add option(s).
Show OpenMP Artifacts

- Select source file, folder, or project
- Run analysis
- See artifacts in OpenMP Artifact view
Show Pragma Region

- Run OpenMP analysis
- Right click on pragma in artifact view
- Select **Show pragma region**
- See highlighted region in C editor
Show OpenMP Problems

- Select **OpenMP problems view**
- Will identify standard OpenMP restrictions
Show Concurrency

- Highlight a statement
- Select the context menu on the highlighted statement, and click **Show concurrency**
- Other statements will be highlighted in yellow
- The yellow highlighted statements *might* execute concurrently to the selected statement
UPC Support

- To see UPC support in C editor, install the optional feature from CDT.
- See Also: http://wiki.eclipse.org/PTP/other_tools_setup#Using_UPC_features

- Filetypes of “upc” will get UPC syntax highlighting, content assist, etc.
- Use preferences to change default for *.c if you like
Remote Development

- PTP already provides the ability to launch/debug remotely
  - However it is often desirable to be able to edit and build remotely
  - If projects are very large, build times may be considerable
- The PTP Remote Development Tools (RDT) will provide a complete remote development environment
  - C/C++ (and Fortran) projects can be hosted on a remote machine
  - Eclipse runs on your local workstation or laptop
  - Files are pulled to local machine only for editing
  - Remote indexing and other services are performed remotely
  - Both managed and Makefile projects are built remotely
  - Uses either Remote System Explorer (RSE) or PTP’s Remote Tools
  - Will have the ability to tunnel over ssh connections
Remote Development (2)

- RDT was introduced with PTP 2.1
  - Configuration is separate from PTP configuration
  - Restricted to RSE connections only (no tunneling)
  - Manual server launch
- PTP 3.0 will seamlessly integrate RDT configuration and simplify setup and use
  - New service model combines PTP and RDT configuration
  - New project wizard has been enhanced and simplified
  - Automatically launch remote server process
  - Still under active development

... So we won’t cover it today
Module 8: Other Tools and Wrap-up

- **Objective**
  - How to find more information on PTP
  - Learn about other tools related to PTP
  - See PTP upcoming features

- **Contents**
  - Links to other tools, including performance tools
  - Planned features for new versions of PTP
  - Additional documentation
  - How to get involved
NCSA HPC Workbench

- Tools for NCSA Blue Waters
  - [http://www.ncsa.illinois.edu/BlueWaters/](http://www.ncsa.illinois.edu/BlueWaters/)
  - Sustained Petaflop system
- Based on Eclipse and PTP
- Includes some related tools
  - Performance tools
  - Scalable debugger
  - Workflow tools ([https://wiki.ncsa.uiuc.edu/display/MRDPUB/MRD+Public+Space+Home+Page](https://wiki.ncsa.uiuc.edu/display/MRDPUB/MRD+Public+Space+Home+Page))
- Part of the enhanced computational environment described at:
  - [http://www.ncsa.illinois.edu/BlueWaters/ece.html](http://www.ncsa.illinois.edu/BlueWaters/ece.html)
NCSA HPC Workbench

Coding & Analysis
(CDT, PLDT, Photran)

PTP Launching & Monitoring

Workflow

Performance Tuning
(HPC toolkit, HPCS toolkit, RENCI, …)

PTP Debugging
PTP-Related Tools

- External Tools Framework
  - Formerly Performance Tools Framework
- Tuning and Analysis Utilities (TAU)
- ISP – In-situ Partial Ordering
  - MPI analysis tools from U.Utah
PTP/External Tools Framework
formerly “Performance Tools Framework”

Goal:

- 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 in the launch configuration window
  - Output is generated, managed and analyzed as specified in the workflow

Module 8
PTP TAU plug-ins

www.cs.uoregon.edu/research/tau/home.php

- TAU (Tuning and Analysis Utilities)
- First implementation of Performance Tools Framework
- Eclipse plug-ins wrap TAU functions, make them available from Eclipse
- Compatible with Photran and CDT projects and with PTP parallel application launching
- Other plug-ins launch Paraprof from Eclipse too
ISP – In-situ Partial Order

- Being contributed to PTP by U. Utah
  - Hope to make available in PTP 3.0 (late Oct.)
- Analyses MPI code dynamically for deadlocks, etc.
- Can match sends and receives
- Can work with several different MPI implementations
ISP – Formal Dynamic Verification of MPI Applications

- Verifies MPI User Applications, generating only the Relevant Process Interleavings
- Detects all Deadlocks, Assert Violations, MPI object leaks, and Default Safety Properties
- Works by Instrumenting MPI Calls Computing Relevant Interleavings, Replaying

(Image courtesy of Steve Parker, U of Utah)
Eclipse CDT/PTP based ISP GUI

ISP Plug-in (trident icon) based on CDT and PTP allows PostVerification Review of Relevant Interleavings, and highlights bugs. It also allows viewing of MPI Happens-Before Relation - a succinct summary of the required MPI orderings.

For details, including Beta download, please visit http://www.cs.utah.edu/formal_verification/ISP-Eclipse
Useful Eclipse Tools

- Python
- Ruby
  - http://sourceforge.net/projects/rubyeclipse
- Subversion (now an Eclipse project)
  - http://eclipse.org/subversive
- Git (now an Eclipse project)
  - http://www.eclipse.org/egit
- ... and many more!
Future PTP Features

- Support for multicore development
  - Building on Cell IDE and other multicore tools
- Resource managers to support for PBS, LSF, and Blue Gene
- Transitioning debugger to Scalable Tools Communication Infrastructure (STCI)
- Enhancements to ETF to support compiler generated reports and optimization directives
- Scalability improvements
  - UI to support 1M processes
  - Optimized communication protocol
  - Very large application support
Information About PTP

- Main web site for downloads, documentation, etc.
  - http://eclipse.org/ptp
- Developers’ wiki for designs, planning, meetings, etc.
  - http://wiki.eclipse.org/PTP
- Articles and other documents:
  - http://wiki.eclipse.org/PTP/articles
Mailing Lists

- **PTP Mailing lists**
  - Major announcements (new releases, etc.) - low volume
    - [http://dev.eclipse.org/mailman/listinfo/ptp-announce](http://dev.eclipse.org/mailman/listinfo/ptp-announce)
  - User discussion and queries - medium volume
    - [http://dev.eclipse.org/mailman/listinfo/ptp-user](http://dev.eclipse.org/mailman/listinfo/ptp-user)
  - Developer discussions - high volume
    - [http://dev.eclipse.org/mailman/listinfo/ptp-dev](http://dev.eclipse.org/mailman/listinfo/ptp-dev)

- **Photran Mailing lists**
  - User discussion and queries
    - [http://dev.eclipse.org/mailman/listinfo/photran](http://dev.eclipse.org/mailman/listinfo/photran)
  - Developer discussions –
    - [http://dev.eclipse.org/mailman/listinfo/photran-dev](http://dev.eclipse.org/mailman/listinfo/photran-dev)
Getting Involved

- See http://eclipse.org/ptp
- Read the developer documentation on the wiki
- Join the mailing lists
- Attend the monthly developer meetings
  - Teleconference each second Tuesday, 1:00 pm ET

- PTP will only succeed with your participation!
PTP Tutorial Feedback

- Please complete feedback form
- Your feedback is valuable!

Thanks for attending
We hope you found it useful