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

July 7, 2010

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.
Eclipse Parallel Tools Platform (PTP)

- Material adapted from SC09 tutorial
- Tutorial source modules indicated in next slide
- All PTP tutorials posted to: [http://wiki.eclipse.org/ptp](http://wiki.eclipse.org/ptp)
- Primary changes: restrict scope to functions that work remotely

- Eclipse 3.6 (Helios) and PTP 4.0 released June 23, 2010
- First coordinated release of Eclipse PTP!
- First release with significant improvement of usability of remote capabilities
## Tutorial Outline

<table>
<thead>
<tr>
<th>Time (Tentative!)</th>
<th>Module</th>
<th>Topics</th>
<th>Presenter</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:30-9:00</td>
<td>1. Overview of Eclipse and PTP</td>
<td>✦ Introduction to Eclipse/PTP</td>
<td>Greg</td>
</tr>
<tr>
<td>9:00-10:00</td>
<td>2. Installation</td>
<td>✦ Prerequisites, Installation</td>
<td>Greg</td>
</tr>
<tr>
<td>10:00-10:30</td>
<td>BREAK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:30-11:15</td>
<td>3. CDT: Working with C/C++</td>
<td>✦ Eclipse basics; Creating a new project ✦ Building and launching</td>
<td>Beth</td>
</tr>
<tr>
<td>11:15-12:00 1:30-2:15</td>
<td>4. Working with MPI (incl. Remote)</td>
<td>✦ CVS, Makefiles, autoconf, PLDT MPI tools ✦ Resource Managers ✦ Launching a parallel application</td>
<td>Jay</td>
</tr>
<tr>
<td>12:00 - 1:30</td>
<td>Lunch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2:15-3:00</td>
<td>5. Debugging</td>
<td>✦ Debugging an MPI program</td>
<td>Greg</td>
</tr>
<tr>
<td>3:00-3:30</td>
<td>BREAK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3:30-4:00</td>
<td>6. Fortran; Refactoring</td>
<td>✦ Photran overview; comparison w/ CDT ✦ Refactoring support</td>
<td>Jeff</td>
</tr>
<tr>
<td>4:00 - 4:45</td>
<td>7. Advanced Features: Performance Tuning &amp; Analysis Tools</td>
<td>✦ PLDT (MPI, OpenMP, UPC tools) (10 min) ✦ TAU, ETFw (15), PPW (5) ✦ ISP (15)</td>
<td>Beth Wyatt/Max Alan</td>
</tr>
<tr>
<td>4:45- 5:00</td>
<td>8. Other Tools, Wrapup</td>
<td>✦ NCSA HPC Workbench, Other Tools, website, mailing lists, future features</td>
<td>Jay/Beth</td>
</tr>
</tbody>
</table>
## Tutorial Outline

<table>
<thead>
<tr>
<th>Time (Tentative!)</th>
<th>Module</th>
<th>Topics</th>
<th>Presenter</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:30-9:00</td>
<td>1. Overview of Eclipse and PTP</td>
<td>✤ Introduction to Eclipse/PTP</td>
<td>Greg</td>
</tr>
<tr>
<td>9:00-10:00</td>
<td>2. Installation</td>
<td>✤ Prerequisites, Installation</td>
<td>Greg</td>
</tr>
<tr>
<td>10:00-10:30</td>
<td>BREAK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:30-11:15</td>
<td>3. CDT: Working with C/C++</td>
<td>✤ Eclipse basics; Creating a new project ✤ Building and launching</td>
<td>Beth</td>
</tr>
<tr>
<td>11:15-12:00</td>
<td>4. Working with MPI (incl. Remote)</td>
<td>✤ CVS, Makefiles, autoconf, PLDT MPI tools ✤ Resource Managers ✤ Launching a parallel application</td>
<td>Jay</td>
</tr>
<tr>
<td>12:00-1:30</td>
<td>Lunch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2:15-3:00</td>
<td>5. Debugging</td>
<td>✤ Debugging an MPI program</td>
<td>Greg</td>
</tr>
<tr>
<td>3:00-3:30</td>
<td>BREAK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3:30-4:00</td>
<td>6. Fortran; Refactoring</td>
<td>✤ Photran overview; comparison w/ CDT ✤ Refactoring support</td>
<td>Jeff</td>
</tr>
<tr>
<td>4:00-4:45</td>
<td>7. Advanced Features: Performance Tuning &amp; Analysis Tools</td>
<td>✤ PLDT (MPI, OpenMP, UPC tools) (10 min) ✤ TAU, ETFw (15), PPW (5) ✤ ISP (15)</td>
<td>Beth Wyatt/Max Alan</td>
</tr>
<tr>
<td>4:45-5:00</td>
<td>8. Other Tools, Wrapup</td>
<td>✤ NCSA HPC Workbench, Other Tools, website, mailing lists, future features</td>
<td>Jay/Beth</td>
</tr>
</tbody>
</table>
Module 3: Working with C/C++, MPI and Remote Machines

Objective

- Learn how to use Eclipse to develop C programs
- Learn how to develop, build and launch, and debug an MPI program on a remote parallel machine

Contents

- Brief introduction to the C/C++ Development Tools (CDT)
- Create a simple application
- Remote project setup
- Working with resource managers
- Launching a parallel application
Workbench

- A Workbench contains perspectives
- A Perspective contains views and editors

- The Workbench represents the desktop development environment
  - Contains a set of tools for resource mgmt
  - Provides a common way of navigating through the resources
- Multiple workbenches can be opened at the same time

Module 3
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.
Switch to C/C++ Perspective

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

- What Perspective am I in? See Title Bar
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
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.)
Preferences

- Eclipse Preferences allow customization of almost everything

- Open Window ➤ Preferences...
- C/C++ preferences allow many options
- Code formatting settings (“Code Style”) shown here
Creating a C/C++ Application

Steps:
- Create a new C project
- Edit source code
- Save and build
Creating a C/C++ Application

Steps:

✧ Create a new C project
✧ Edit source code
✧ Save and build

Note: the next several slides will illustrate some local development concepts. While local development is not the focus of this mini-tutorial, we will use what we learn to help with remote development.
Create a new C 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 **Finish**
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
Double-click on source file in the **Project Explorer** to open C editor.

Outline view is shown for file in editor.

We’ll describe the editor in the next few slides...
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.
- How to Save
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
  - Task Tags, Bookmarks
- Location bar for navigating to interesting features in the entire file

Icons:

- Task tag
- Warning
- Error
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
Build

- Your program should build when created.
- To rebuild, many ways include:
  - Select project, Hit hammer icon in toolbar
  - Select project, Project ➤ Build Project
  - Right mouse on project, Clean Project
Build (2)

- See the results of the build in the Console View
- Executable should be in Debug folder:

![Console View](image)

```bash
make all
Building file: ../src/MyHelloProject.c
Invoking: GCC C Compiler
gcc -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"src/MyHelloProject.d" -MT"src/MyHelloProject.d" -o"src/MyHelloProject.o" ' ../src/MyHelloProject.c'
Finished building: ../src/MyHelloProject.c

Building target: MyHelloProject
Invoking: GCC C Linker
gcc -o"MyHelloProject" ./src/MyHelloProject.o
Finished building target: MyHelloProject
```
Here, introduce the remote project, remote build
Preparation for Remote C/C++ Project

General notes:
- This has been tested and found to work on the following machines: Queenbee
- The example shown is worked on Queenbee

There are “out of band” setup steps that you need to do prior to starting the tutorial
- Copy sample MPI code to your directory
- Set up environment on remote machine
- Please refer to setup instructions for your machine
Remote C/C++ Project

❖ While in the C/C++ perspective:
   ❖ Right click in the Project Explorer
   ❖ Select **New > Remote C/C++ Project**
Remote Project Wizard

This starts the New Remote Project Wizard
Remote Project Wizard

- This starts the **New Remote Project Wizard**
- Name the project
Remote Project Wizard

- This starts the **New Remote Project Wizard**
- Name the project
- Select **Remote Tools** as the Remote Provider
- Select **New**… Connection
New Remote Connection

- Remote Host Configuration
  - Name the remote target

#### Target Environment Configuration

**Remote Host**

- Properties for connecting to a remote host

**Host Information**

- **Target name**: queenbee-tgtn01
- **Host**: login1-qb.loni-isu.teragrid.org
- **Port**: 22
- **User**: tgtn01
- **Password based authentication**
- **Password**: ********

**Advanced**

- **File with private key**: Browse
- **Passphrase**:

**Finish**  **Cancel**
New Remote Connection

- Remote Host Configuration
  - Name the remote target
  - Select Remote Host, and fill in the hostname
    - Use the host you were assigned

Module 3
New Remote Connection

- Remote Host Configuration
  - Name the remote target
  - Select Remote Host, and fill in the hostname
    - Use the host you were assigned
  - And add in your username and password (this is currently not editable once saved)
Finish Remote Setup

- Choose new remote connection
- Then select **Browse** to choose one’s working directory
- N.B. this is conceptually similar to “importing” existing code into a project, note that you can also create an empty directory and start coding away!
Finish Remote Setup

- Choose new remote connection
- Then select **Browse** to choose one’s working directory
  - N.B. this is conceptually similar to “importing” existing code into a project, note that you can also create an empty directory and start coding away!
- Choose the *mpi* sample code directory, and then select OK
Finish Remote Setup

- We’ll step through the rest of the wizard to finish the remote project
- Select **Next**
Finish Remote Setup

- We’ll step through the rest of the wizard to finish the remote project
  - Select **Next**
  - And **Next**
Finish Remote Setup

+ We’ll step through the rest of the wizard to finish the remote project
  + Select **Next**
  + And **Next**
  + And **Finish**

+N.B. We’ll be creating a new service configuration
Finish Remote Setup

- We’ll step through the rest of the wizard to finish the remote project
  - Select **Next**
  - And **Next**
  - And **Finish**
  - N.B. We’ll be creating a new service configuration
  - Select **OK** to override existing project settings
Remote C/C++ Project

✦ If prompted to switch to the Remote C/C++ perspective, select **OK**.
Remote C/C++ Project

- If prompted to switch to the Remote C/C++ perspective, select **OK**.
Remote C/C++ Project

- If prompted to switch to the Remote C/C++ perspective, select **OK**.
- Note that there is an issue with the ring.c source.
Remote C/C++ Project

- If prompted to switch to the Remote C/C++ perspective, select **OK**.
  - Note that there is an issue with the ring.c source.
  - Double click on `ring.c` to load it into the editor, to see if we can determine the issue.
Ring.c

Note the header files marked in red
Ring.c

- Note the header files marked in red.
- Also, note that the remote indexer has produced a source outline.
Note the header files marked in red

Also, note that the remote indexer has produced a source outline

We need to amend the include paths for this project...

Module 3
Include paths

- Right click on project name, then select Properties
Project properties

- Open **Remote Development**,  
- Select **Remote Paths and Symbols**
Open **Remote Development**, **Select Remote Path and Symbols**, **Click Add**.

Module 3
Remote Include Directory

- Add /usr/include for queenbee
  - Note to check “add to all configurations” and “add to all languages
  - Hit OK
Remote Include Directory

- Add /usr/include for queenbee
  - Note to check "add to all configurations" and "add to all languages"
  - Hit OK
- Repeat for MPI includes...
Remote include paths

- And you have the paths you need for your platform
- Note that the details differ by platform
- Press “OK” to finalize the changes
Remote include paths

- And you have the paths you need for your platform
  - Note that the details differ by platform
  - Press “OK” to finalize the changes
- Allow Eclipse to rebuild the index by pressing “Yes”
Include path issues resolved

- Verify that include paths are now resolved
Include path issues resolved

- Verify that include paths are now resolved
- Also note the outline view
Follow an include path...

- Right-click on mpi.h, and select **Open Declaration**
Following declarations

And you can see the contents of mpi.h on your system.
Following declarations

- And you can see the contents of mpi.h on your system.

Note outline view changed as well.
Let’s start our build by doing a clean build

Project>Clean...
Let’s start our build by doing a clean build

- **Project>Clean...**
- Select the project we are working on (and the clean projects selected below button)
- Press OK
Build clean

- Note output from compiler/make
Build clean

- Note output from compiler/make
- Ring, ring.o should also appear in your project

Module 3
Build clean

- Note you can refresh your project’s file list – right click on Project Explorer, then Refresh
- You can also refresh specific projects, by right-clicking on the project itself
Build problems?

- If there are problems, see:
  - Marker on editor line
  - **Problems view**
  - Double-click on line in **Problems** view to go to location of error
Build problems? Try it

- Remove a semicolon from a line in your “Hello World” example
- Save file
- Rebuild
- **See the Problems view**
- Double-click on line in **Problems** view to go to location of error
- Fix it and rebuild to continue

*Module 3*
Running your code

- Switch to the parallel runtime perspective
- **Window** > **Open Perspective** > **Other...**
Running your code

- Switch to the parallel runtime perspective
- Window > Open Perspective > Other...
- Choose Parallel Runtime
- Then press OK
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
PTP Runtime Perspective

- Resource managers view
- Machines view
- Node details view
- Jobs view
- Console view
- Properties view

Module 4
Add a resource manager

- Right click in resource managers view, select **Add Resource Manager**...
Add resource manager

- We’ll add 2 resource managers for queenbee – one for OpenMPI, another for PBS
Add resource manager

✦ We’ll add 2 resource managers for queenbee – one for OpenMPI, another for PBS
✦ Choose OpenMPI, then select Next
Open MPI Resource Manager

- Select **Remote Tools** as the remote service provider

![Remote Tools configuration window](image)
OpenMPI Resource Manager

- Select **Remote Tools** as the remote service provider
- Then select the appropriate connection name
OpenMPI Resource Manager

- Select **Remote Tools** as the remote service provider
- Then select the appropriate connection name
- **Important:** be sure to click **SSH port forwarding** to enable ssh tunneling of connections back to your laptop
OpenMPI wizard

Leave this page as default values, select next
OpenMPI wizard

- Leave this page as default values, select **next**
- Only start resource manager when you need it (leave Startup unselected)
- Press **Next**
OpenMPI wizard

- Leave this page as default values, select **next**
- Only start resource manager when you need it (leave Startup unselected)
- Press **Next**
- Add to an existing service configuration
- Select the machine we were working on, then select **Finish**
Repeat for the PBS Resource Manager

- Right-click in the Resource Manager view, select **Add Resource Manager...** again
PBS Resource Manager

- Select PBS Resource Manager... then Next>
PBS Resource Manager

- Select PBS Resource Manager... then Next>
- Select Remote Tools Remote Service Provider
PBS Resource Manager

- Select PBS Resource Manager... then Next>
- Select Remote Tools Remote Service Provider
- And appropriate connection name
PBS Resource Manager

- Select PBS Resource Manager... then Next>
- Select **Remote Tools** Remote Service Provider
- And appropriate connection name
- And enable SSH port forwarding
PBS Resource Manager

- Leave the PBS Batch Script Configuration as default
  - Provides an opportunity to customize resource manager
- Select **Next**
PBS Resource Manager

- Leave the PBS Batch Script Configuration as default
- Provides an opportunity to customize resource manager
- Select **Next**
- Only start resource manager manually
- And select **Next**
This time, create a new service configuration
And select Finish
Resource managers

Voila! We now have two resource managers
Resource managers

Voila! We now have two resource managers.
Resource managers

✧ Voila! We now have two resource managers.

✧ To start a resource manager – right click on the resource manager (OpenMPI this time), and select Start Resource Manager.
OpenMPI Resource Manager

- Note that Resource Manager is now running (green icon)
OpenMPI Resource Manager

- Note that Resource Manager is now running (green icon)
- Also, queenbee as a system is denoted as up and running
OpenMPI Resource Manager

- Note that Resource Manager is now running (green icon)
- Also, queenbee as a system is denoted as up and running
- And we have one head node available for interactive use
To run our code, select Run > Run Configurations
Run Configuration

✦ Sneaky detail –
✦ Select **Parallel Application**, then
Run Configuration

✦ Sneaky detail –
✦ Select **Parallel Application**, then
✦ Select the **New** button
Run Configuration

- Sneaky detail –
  - Select **Parallel Application**, then
  - Select the **New** button
Run Configuration

- Sneaky detail –
  - Select **Parallel Application**, then
  - Select the **New** button
- Now need to setup the Resource Manager (pull down resource manager menu)
Run Configuration

- Sneaky detail –
  - Select **Parallel Application**, then
  - Select the **New** button
- Now need to setup the Resource Manager (pull down resource manager menu)
  - Choose OpenMPI
OpenMPI Configuration

- Note that the OpenMPI has both basic and advanced attributes...
  - We’ll stick with basic for today
OpenMPI Configuration

- Note that the OpenMPI has both basic and advanced attributes...
  - We’ll stick with basic for today
- Configure 2 processes
  - And allocate “by slot”
OpenMPI Configuration

- Note that the OpenMPI has both basic and advanced attributes...
  - We’ll stick with basic for today
- Configure 2 processes
  - And allocate “by slot”
- Select the **Application** tab
OpenMPI Configuration

- **Browse** for the application code
OpenMPI Configuration

- **Browse** for the application code
- Select the ring executable we built previously, then press **OK**
OpenMPI Configuration

- We'll need to configure the debugger as well.
- Select debugger tab.
OpenMPI Configuration

- We’ll need to configure the debugger as well
  - Select debugger tab
  - Select **SDM** debugger
We'll need to configure the debugger as well:

- Select debugger tab
- Select **SDM** debugger
- Browse to the location of the executables
We’ll need to configure the debugger as well:

- Select debugger tab
- Select SDM debugger
- Browse to the location of the executables
- Press Apply
We’ll need to configure the debugger as well:
- Select debugger tab
- Select SDM debugger
- Browse to the location of the executables

Press **Apply**, then press **Run**
Interactive code run

- Note successful completion (refer to icons previously)
Interactive code run

- Note successful completion (refer to icons previously)
- Code output to console
- Click on node to get process information
Process information

✦ Note information on the 2 MPI ranks
Process information

Note information on the 2 MPI ranks
Process information

* Note information on the 2 MPI ranks
Process information

- Note information on the 2 MPI ranks
  - Rank 1
  - Rank 1 output
Process information

- Note information on the 2 MPI ranks
  - Rank 1
  - Rank 1 output
  - Rank 0
  - Rank 0 output
Module 5: Parallel Debugging

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

- **Contents**
  - Launching a parallel debug session
  - The PTP Debug Perspective
  - 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
- First, select **Debug Configurations...** to verify the debugger settings
Verify the Debugger Tab

- Select **Debugger** tab
- Make sure **SDM** is selected in the **Debugger** dropdown
- Use the **Browse** button to select the debugger executable if required
  - If launching remotely, the debugger executable must also be located remotely
- Debugger session address should not need to be changed
- Click on **Debug** to launch the program
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.
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.

![Image](image-url)
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.
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 from process %d!
```

Module 5
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.
Other CDT features

- Searching
- Open Declaration / hyperlinking between files in the editor
- Rename in file (in-place in editor)
- Refactoring
  - Rename refactoring / Preview panes
  - Extract Constant refactoring
  - Other refactorings in CDT
Language-Based Searching

- “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
Open Declaration
Need this for C

- 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”
In Java (Murphy-Hill et al., ICSE 2008):

- **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**
- 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
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.
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)
Online Information

✦ 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

✦ Information about Photran
  ✦ Main web site for downloads, documentation, etc.
    ✦ http://eclipse.org/photran
  ✦ User’s manuals
    ✦ http://wiki.eclipse.org/PTP/photran/documentation
Mailing Lists

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

- **Photran Mailing lists**
  - User discussion and queries
    - http://dev.eclipse.org/mailman/listinfo/photran
  - Developer discussions –
    - 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!