Developing Scientific Applications with the IBM Parallel Environment Developer Edition

Greg Watson, IBM
grw@us.ibm.com

Christoph Pospiech, IBM
christoph.pospiech@de.ibm.com

ScicomP ’13
May 2013
Tutorial Highlights

- Provides a brief introduction to Eclipse
- Shows how to import existing code into Eclipse
- Shows how to build and run an MPI application
- Shows features of PE Developer Environment:
  - Monitoring hardware performance counters
  - Profiling/tracing and MPI application
  - Profiling I/O activity in an application
- Shows debugging with the IBM Parallel Debugger
<table>
<thead>
<tr>
<th>Time</th>
<th>Module</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:30 – 14:00</td>
<td>1. Overview of Eclipse and PTP</td>
<td>+ Introduction to Eclipse/PTP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ Installation of Parallel Package</td>
</tr>
<tr>
<td>14:00 – 15:00</td>
<td>2. Eclipse Basics</td>
<td>+ Importing a project</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ Editing, building, launching</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ Working with MPI, Fortran</td>
</tr>
<tr>
<td>15:00 – 15:30</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>15:30 – 16:00</td>
<td>3. Advanced Features</td>
<td>+ Target system configurations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ Advanced development and refactoring</td>
</tr>
<tr>
<td>16:00 – 16:30</td>
<td>4. IBM HPC Toolkit</td>
<td>+ Profile application using Xprof</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ Hardware performance counters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ MPI Profiling and Tracing</td>
</tr>
<tr>
<td>16:30 – 16:55</td>
<td>5. IBM Parallel Debugger</td>
<td>+ Overview of features</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ Hands-on workshop exercises</td>
</tr>
<tr>
<td>16:55 – 17:00</td>
<td>6. Wrap-up</td>
<td>+ Overview of features not covered</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ Resources, getting questions answered</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ How to participate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ Participant feedback</td>
</tr>
</tbody>
</table>
Installation

Objective
- To learn how to install Eclipse and PTP

Contents
- System Prerequisites
- Download and Install Eclipse
System Prerequisites

- Local system (running Eclipse)
  - Linux (just about any version)
  - MacOSX (10.5 Leopard or higher)
  - Windows (XP on)
- Java: Eclipse requires Sun or IBM Java
  - Only need Java runtime environment (JRE)
  - Java 1.6 or higher
    - Java 1.6 is the same as JRE 6.0
  - The GNU Java Compiler (GCJ), which comes standard on Linux, will not work!
  - OpenJDK, distributed with some Linux distributions, has not been tested by us but should work.
  - See http://wiki.eclipse.org/PTP/installjava
Eclipse Packages

- The current version of Eclipse (4.2) is also known as “Juno”
  - Next release “Kepler” in June 2013
- Eclipse is available in a number of different packages for different kinds of development
  - http://eclipse.org/downloads
- For PTP, we recommend the all-in-one download:
  - Eclipse for Parallel Application Developers

We often call this the “Parallel Package”
Pre-release Package

- For this tutorial we will be using a pre-release version of Eclipse “Kepler”
- This version will be released at the end of June
- The Parallel Package is available from the “Developer Builds” link on the main downloads page
  - [http://eclipse.org/downloads](http://eclipse.org/downloads)
Exercise

1. Go to the “Developer Builds” page of the main Eclipse download site
2. Download the “Eclipse for Parallel Application Developers” package to your laptop
   ✦ Your tutorial instructions will provide the location of the package
   ✦ Make sure you match the architecture with that of your laptop
3. If your machine is Linux or Mac OS X, untar the file
   ✦ On Mac OS X you can just double-click in the Finder
4. If your machine is Windows, unzip the file
5. This creates an eclipse folder containing the executable as well as other support files and folders
Starting Eclipse

- **Linux**
  - From a terminal window, enter
    “<eclipse_installation_path>/eclipse/eclipse &”

- **Mac OS 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
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 default workspace location is fine for this tutorial

The prompt can be turned off.
Eclipse Welcome Page

- Displayed when Eclipse is run for the first time
  Select “Go to the workbench”
Introduction

Objective
- To introduce the Eclipse platform and PTP

Contents
- New and Improved Features
- 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 Features

- Full development lifecycle support
- Revision control integration (CVS, SVN, Git)
- Project dependency management
- Incremental building
- Content assistance
- Context sensitive help
- Language sensitive searching
- Multi-language support
- Debugging
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, UPC, etc.)
- Support for the integration of parallel tools
- An environment that simplifies the end-user interaction with parallel systems

http://www.eclipse.org/ptp
How Eclipse is Used

Introduction

Local Source Code

Remote Source Code

Edit/Build
Launch/Monitor
Debugging
Performance Tuning
## Parallel Environment Developer Edition 1.3

### PE DE Client

<table>
<thead>
<tr>
<th>Programming models</th>
<th>HPC Toolkit</th>
<th>Parallel Tools Platform</th>
<th>Base Tools Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>• MPI</td>
<td>• MPI profiling</td>
<td>• PTP 7.0</td>
<td>Eclipse Platform (4.3)</td>
</tr>
<tr>
<td>• LAPI</td>
<td>• OpenMP profiling*</td>
<td>• Parallel Language Development Tools</td>
<td>C/C++ Development Tools (CDT 8.2)</td>
</tr>
<tr>
<td>• PAMI</td>
<td>• Hardware counter profiling</td>
<td>• Parallel application launch</td>
<td>Fortran Development Tools (Photran 8.1)</td>
</tr>
<tr>
<td>• UPC</td>
<td>• I/O profiling</td>
<td>• System monitoring</td>
<td>Linux Tools</td>
</tr>
<tr>
<td>• OpenMP</td>
<td>• Profiling and trace visualization</td>
<td>• Synchronized projects</td>
<td></td>
</tr>
<tr>
<td>• OpenSHMEM</td>
<td>• Binary instrumentation*</td>
<td>• Remote projects</td>
<td></td>
</tr>
</tbody>
</table>

* Supported on Power only

### PE DE Server

<table>
<thead>
<tr>
<th>HPC Toolkit</th>
<th>Instrumentation Libraries</th>
<th>Command-line Tools</th>
<th>Operating Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peekperf Visualization</td>
<td>Xprof Visualization</td>
<td></td>
<td>Windows</td>
</tr>
</tbody>
</table>

### PE RE

<table>
<thead>
<tr>
<th>Scalable Communication Infrastructure (SCI)</th>
<th>POE Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadLeveler</td>
<td>LSF</td>
</tr>
<tr>
<td>Power</td>
<td>x86</td>
</tr>
</tbody>
</table>

| BG/Q |

### Operating Systems

- **Windows**
- **Linux**
- **Mac OS X**
- **AIX**

**Supported on Power only**

[http://ibm.co/12L2RxA](http://ibm.co/12L2RxA)
Importing a Project

Objective
- Learn how to import a project into Eclipse
- Learn how to manage synchronized projects

Contents
- Eclipse project locations
- Creating a synchronized project
- Managing synchronized project properties
- Using synchronize filters
Project Location

- **Local**
  - Source is located on local machine, builds happen locally
  - This is the default Eclipse model

- **Synchronized**
  - Source is located on both local and remote machine(s), then kept in synchronization by Eclipse
  - Building and launching happens remotely (can also happen locally)
  - Used mainly for scientific and supercomputing applications

- There are also remote-only projects, but these have limitations and are not covered here
Synchronized Projects

Projects types can be:

- File Service
- Index Service
- Launch Service
- Build Service
- Debug Service
- Local source code copy
- Remote source code copy

Local | Remote

Sync-2
Revision Control Systems (RCS)

- Eclipse supports a range of RCS, such as CVS, Git, and Subversion (and others)
- These are distinct from synchronized projects
- RCS can be used in conjunction with synchronized projects
- Synchronized projects are typically *not* used for revision control
Synchronized Project Creation

- **Local -> Remote**
  - Projects start out local then are synchronized to a remote machine
  - Three options
    - Created from scratch
    - Imported from local filesystem
    - Imported from RCS

- **Remote -> Local**
  - Projects start out on remote machine then are synchronized to the local system
  - Two options
    - Already on remote system
    - Checked out from RCS
Source Code for project

- Source code exists on remote target

```
$ pwd
/gpfs/ikmu/tibbitts/shallow
$ ls -l

total 2880
drwxr-xr-x 2 tibbitts users 32768 Mar 16 15:53 .
drwxr-xr-x 7 tibbitts users 32768 Mar 15 18:38 ..
-rw-r--r-- 1 tibbitts users 1741 Feb 11 16:25 calc.c
-rw-r--r-- 1 tibbitts users 2193 Feb 11 16:25 copy.c
-rw-r--r-- 1 tibbitts users 2873 Jan 25 08:52 decs.h
-rw-r--r-- 1 tibbitts users 2306 Feb 11 16:25 diag.c
-rw-r--r-- 1 tibbitts users 2380 Feb 11 16:25 dump.c
-rw-r--r-- 1 tibbitts users 2512 Feb 11 16:25 init.c
-rw-r--r-- 1 tibbitts users 6161 Mar 15 19:27 main.c
-rw-r--r-- 1 tibbitts users 718 Mar 15 18:34 Makefile
-rw-r--r-- 1 tibbitts users 1839 Feb 11 16:25 time.c
-rw-r--r-- 1 tibbitts users 2194 Feb 11 16:25 tstep.c
-rw-r--r-- 1 tibbitts users 8505 Feb 11 16:25 worker.c
$  ```
Create Synchronized Project

- In the Project Explorer, right click then choose
  - **New>Synchronized C/C++ Project** if your project is C/C++ only
  - **New>Synchronized Fortran Project** if your project contains Fortran files

- This adds a Fortran nature so you can access Fortran properties, etc.
New Synchronized Project Wizard

- Enter the **Project Name**
  - E.g. “shallow”
- The **Local Directory** specifies where the local files are located
  - Leave as default
- The **Remote Directory** specifies where the remote files are located
  - Select a connection to the remote machine, or click on **New**... to create a new one (See next slide)
  - Browse for the directory on the remote machine
- Use **Modify File Filtering**... if required (see later slide)

*Synchronized Projects*
Creating a Connection

- In the **Target Environment Configuration** dialog
  - Enter a **Target name** for the remote host
  - Enter host name, user name, and user password or other credentials
  - Select **Finish**
Choose the **Project Type**
- If you are synchronizing with an existing project, use **Makefile Project>Empty Project**
- Otherwise, choose the type of project you want to create

Choose the toolchain for the remote build
- Use a toolchain that most closely matches the remote system

Choose a toolchain for the local build
- This is used for advanced editing/searching

Click **Finish** to create the project
Synchronized Project Menu

- Synchronized projects are indicated with a “synchronized” icon
- Right click on project to access **Synchronize** menu
  - **Sync Active Now** will manually synchronize the active configuration
  - **Set Active** can be used to select the active configuration
  - **Manage…** is used to create new configurations to synchronize to different target systems
  - **Sync All Now** will manually synchronizes all configurations

- **Auto-Sync (Global)** will enable or disable automatic synchronization
- **Auto-Sync Settings** can be used to select which configurations will be synchronized
- **Filter…** is used to change the filter settings for the project
Manage Configurations

- Used to manage synchronize configurations
- Use **Set Active** to change the active configuration (shown in **bold** in the list of configurations)
- Use **Add** to add a new configuration in order to synchronize to a different target system
- Other configuration information, such as the default build configuration, can also be changed

**By default, there will be a configuration for the target system (active) and a Local configuration. The Local configuration can be used to build a local copy of the project if desired.**

Synchronized Projects
Synchronize Filters

- If not all files in the remote project should be synchronized, a filter can be set up
  - For example, it may not be desirable to synchronize binary files, or large data files
- Filters can be created at the same time as the project is created
  - Click on the **Modify File Filtering...** button in the New Project wizard
- Filters can be added later
  - Right click on the project and select **Synchronizee>Filter...**
Synchronize Filter Dialog

- Files can be filtered individually by selecting/unselecting them in the **File View**
- Include or exclude files based on:
  - Paths
  - Regular expressions
  - Wildcards (foo.*)
Synchronized Project Properties

- Synchronized configurations can also be managed through the project properties
- Open the project properties by right-clicking on the project and selecting Properties
  - Select Synchronize
- This is the same as using the Synchronize>Manage... menu
Forcing a Resync

- If Auto-sync is set, the project should automatically resync with remote system when things change.
- Sometimes you may need to do it explicitly.
- Right click on project and select Synchronization > Sync Active Now.
- Status area in lower right shows when Synchronization occurs.
Remote Terminal

- There is a remote terminal that can be used to provide a shell from within Eclipse
- Select **Window>**Show **View>**Other...
- Choose **Terminal** from the Terminal folder
- In the **Terminal** view, click on the **New Terminal Connection** button
- Choose a previously configured connection from the dropdown, or create a new one
Changing Remote Connection Information

- If you need to change remote connection information (such as username or password), use the **Remote Environments** view.

- Stop the remote connection first.

- Right-click and select **Edit**.

- Note: Remote Host may be stopped.
  - Any remote interaction starts it.
  - No need to restart it explicitly.

*Sync-17*
Exercise

1. Import an existing project into a synchronized project
   ✦ Your login information and source directory will be provided by the tutorial instructor
2. Observe that the project files are copied to your workspace
3. Open a file in an editor, add a comment, and save the file
4. Observe that the file is synchronized when you save the file
   ✦ Watch lower-right status area; confirm on host system

Synchronized Projects Sync-18
Optional Exercise

1. Experiment with sync filters
   ✷ Create a filter to exclude files ending in “.o”
   ✷ Create an empty file ending in “.o” on the local system and verify it is not copied to remote machine after a sync
   ✷ Try creating a file on the remote system and verify that it is not copied

2. Create a new synchronize configuration
   ✷ Use the same host, different directory
   ✷ Synchronize the new configuration and verify that you have a second copy on the remote machine
Building a Project

Objective
- Learn how to build an MPI program on a remote system

Contents
- How to change build settings
- How to start a build and view build output
- How to clean and rebuild a project
- How to create build targets
Build Configurations

- A build configuration provides the necessary information to build the project.
- The build configuration information is specified in the project properties.
- Projects can have multiple build configurations, each configuration specifies a different set of options for a build.
- Open the properties by right-clicking on the project name in the Project Explorer view and selecting Properties (bottom of the context menu list).

Note: Fortran projects are a superset of C/C++ projects, so they have properties for both.
Build Properties (1)

- **C/C++ Build**
  - Main properties page
  - Configure the build command
  - Default is “make” but this can be changed to anything

- **Build Variables**
  - Create/manage variables that can be used in other build configuration pages

- **Discovery Options**
  - Obsolete options page for discovering compiler settings (use **C/C++ General>Preprocessor Include Paths instead**)

- **Environment**
  - Modify/add environment variables passed to build

- **Logging**
  - Enable/disable build logging
Build Properties (2)

- **Settings**
  - Binary parser selection (used to display binaries in Project Explorer)
  - Error parser selection (used to parse the output from compiler commands)
  - Tool Chain settings (managed projects only)

- **Tool Chain Editor**
  - Allows the tools in a particular tool chain to be modified

- **XL C/C++ Compiler**
  - Compiler settings for XL C/C++ compilers (if installed)
Selecting Build Configuration

- Multiple build configurations may be available
  - Synchronized projects will usually have a remote and a local build configuration
  - Build configurations for different architectures

- The active build configuration will be used when the build button is selected

- The **Build Configurations** project context menu can be used to change the active configuration
  - Right click on project, then select the build configuration from the **Build Configurations > Set Active** menu

![Build Configurations Menu](image-url)
Building Synchronized Projects

- When the build button is selected, the "active" build configuration will be built on the remote system specified by the "active" synchronize configuration.
- The build and synchronize configurations are independent.
  - It is possible to change which build configuration is active, but make sure this makes sense on the remote system specified in the synchronize configuration.
- A build configuration can be associated with a synchronize configuration, so that it is automatically selected when the synchronize configuration is changed.
Configuring the Build Environment

- If the remote system has an environment system (such as Modules) installed, a custom set of modules can be configured for building C/C++ projects.
- In the Manage Synchronize Configurations dialog, select the configuration you wish to change.
- Check Use an environment management system to customize the remote build environment.
Select a module from the **Available Modules** list and click the **Add->** button to add them to the **Selected Modules** list.

Use the **<-Remove** button to remove modules from the Selected Modules list.

Use the **Filter list** field to quickly find modules with a given name.

Use the **Up** and **Down** buttons to change the order of the **Selected Modules**.

Click **Select Defaults** to load only those modules that are present in a new login shell.
To build the project, Eclipse will
- Open a new Bash login shell
- Execute `module purge`
- Execute `module load` for each selected module
- Run `make`

Module commands are displayed in the Console view during build
- Beware of modules that must be loaded in a particular order, or that contain common paths like /bin or /usr/bin
For this tutorial, we want to use gcc and Open MPI

- Navigate to gnu/4.1.2 in **Available Modules** and select **Add ->**
- Navigate to openmpi/1.4.3 and select **Add ->**
Starting the Build

- Select the project in Project Explorer

- Click on the hammer button in toolbar to run a build using the active build configuration
Viewing the Build Output

- Build output will be visible in console

![Console output of build process](image)
Build Problems

- Build problems will be shown in a variety of ways
  - Marker on file
  - Marker on editor line
  - Line is highlighted
  - Marker on overview ruler
  - Listed in the **Problems view**

- Double-click on line in **Problems view** to go to location of error in the editor
Forcing a Rebuild

* If no changes have been made, make doesn’t think a build is needed e.g. if you only change the Makefile

* In **Project Explorer**, right click on project and select **Clean Project**

* Build console will display results of clean

* Rebuild project by clicking on build button again
Creating Make Targets

- By default
  - The build button will run “make all”
  - Cleaning a project will run “make clean”
- Sometimes, other build targets are required
- Open Make Target view
- Select project and click on New Make Target button
- Enter new target name
- Modify build command if desired
- New target will appear in view
- Double click on target to activate

Building a Project
Exercise

1. Start with your ‘shallow’ project
2. Set the gnu/4.1.2 and openmpi/1.4.3 modules
3. Build the project
4. Edit a source file and introduce a compile error
   ✦ In main.c, line 97, change ‘;’ to ‘:’
   ✦ Save, rebuild, and watch the Console view
   ✦ Use the Problems view to locate the error
   ✦ Locate the error in the source code by double clicking on the error in the Problems view
   ✦ Fix the error
5. Rebuild the project and verify there are no build errors
Optional Exercises

1. Open the Makefile in Eclipse. Note the line starting with “tags:” – this defines a make target named **tags**.

2. Open the Outline view while the Makefile is open. What icon is used to denote make targets in the Outline?

3. Right-click the **tags** entry in the Outline view. Add a Make Target for **tags**.

4. Open the Make Targets view, and build the **tags** target.

---

Building a Project

1. Rename Makefile to Makefile.mk

2. Attempt to build the project; it will fail

3. In the project properties (under the C/C++ Build category), change the build command to: `make -f Makefile.mk`

4. Build the project; it should succeed
Running an Application

Objective
- Learn how to run an MPI program on a remote system

Contents
- Creating a run configuration
- Configuring the application run
- Monitoring the system and jobs
- Controlling jobs
- Obtaining job output
Creating a Run Configuration

- Open the run configuration dialog **Run>Run Configurations**...
- Select **Parallel Application**
- Select the **New** button

Or, just double-click on **Parallel Application** to create a new one

**Note:** We use “Launch Configuration” as a generic term to refer to either a “Run Configuration” or a “Debug Configuration”, which is used for debugging.
Set Run Configuration Name

- Enter a name for this run configuration
  - E.g. “shallow”
- This allows you to easily re-run the same application
- If the “shallow” project was selected when the dialog was opened, its name will be automatically entered
In Resources tab, select a Target System Configuration that corresponds to your target system

- Use edu.sdsc.trestles.torque.batch

Target system configurations can be generic or can be specific to a particular system

- Use the specific configuration if available, or the generic configuration that most closely matches your system

- You can type text in the box to filter the configurations in the list

Running an Application
Configure the Connection

- Choose a connection to use to communicate with the target system
- If no connection has been configured, click on the New button to create a new one
  - Fill in connection information, then click ok
- The new connection should appear in the dropdown list
- Select the connection you already have to trestles.sdsc.edu
- Select toggle if you don’t want to see popup again
The content of the **Resources** tab will vary depending on the target system configuration selected.

This example shows the TORQUE configuration.

For TORQUE, you will normally need to select the **Queue** and the **Number of nodes**.

For parallel jobs, choose the **MPI Command** and the **MPI Number of Processes**.

---

**Running an Application**
Resources Tab (2)

- For this tutorial, use the following values
  - Queue: shared
  - Number of nodes: 1:ppn=5
  - MPI Command: mpirun
  - MPI Number of Processes: 5
  - Leave other fields alone

- Configure modules for running the application
  - Click on the Modules to Load: Configure... button
  - Select the same modules used to build
    - gnu/4.1.2
    - openmpi/1.4.3
Viewing the Job Script

- Some target configurations will provide a **View Script** button.
- Click on this to view the job script that will be submitted to the job scheduler.
- Batch scheduler configurations should also provide a means of importing a batch script.

---

Running an Application
Select the **Application** tab

Choose the **Application program** by clicking the **Browse** button and locating the executable on the remote machine

- Use the same “shallow” executable

Select **Display output from all processes in a console view**
Arguments Tab (Optional)

- The **Arguments** tab lets you supply command-line arguments to the application.
- You can also change the default working directory when the application executes.

*Running an Application*
The **Environment** tab lets you set environment variables that are passed to the job submission command. This is independent of the Environment Management (module/softenv) support described in a separate module.
Synchronize Tab (Optional)

- The **Synchronize** tab lets you specify upload/download rules that are execute prior to, and after the job execution.
- Click on the **New upload/download rule** buttons to define rules.
- The rule defines which file will be uploaded/downloaded and where it will be put.
- Can be used in conjunction with program arguments to supply input data to the application.

*Running an Application*
Common Tab (Optional)

- The **Common** tab is available for most launch configuration types (not just Parallel Application)
- Allows the launch configuration to be exported to an external file
- Can add the launch configuration to the favorites menu, which is available on the main Eclipse toolbar
- Select **Run** to launch the job
Run

- Select **Run** to launch the job
- You may be asked to switch to the System Monitoring Perspective

- Select **Remember my decision** so you won’t be asked again
- Select **Yes** to switch and launch the job
System Monitoring Perspective

- System view
- Jobs running on system
- Active jobs
- Inactive jobs
- Messages
- Console

Scroll to see more

Running an Application
Moving views

- The System Monitoring Perspective overlaps the **Active Jobs** and **Inactive Jobs** views.
- To split them apart and see both at once, **drag** the tab for the **Inactive Jobs** view to the lower half of its area, and let go of mouse.
System Monitoring

- **System** view, with abstraction of system configuration
- Hold mouse button down on a job in **Active Jobs** view to see where it is running in **System** view
- Hover over node in **System** view to see job running on node in **Active Jobs** view

Running an Application

One node with 16 cores
Job Monitoring

- Job initially appears in **Inactive Jobs** view
- Moves to the **Active Jobs** view when execution begins
- Returns to **Inactive Jobs** view on completion
- Status refreshes automatically every 60 sec
- Can force refresh with menu

*Running an Application*
Controlling Jobs

- Right click on a job to open context menu
- Actions will be enabled IFF
  - The job belongs to you
  - The action is available on the target system
  - The job is in the correct state for the action
- When job has COMPLETED, it will remain in the Inactive Jobs view
After status changes to COMPLETED, the output is available:
- Right-click on the job
- Select **Get Job Output** to display output sent to standard output
- Select **Get Job Error** to retrieve output sent to standard error

Output/Error info shows in Console View.

Jobs can be removed by selecting **Remove Job Entry**.

*Running an Application*
Add a Monitor

- You can monitor other systems too
- In **Monitors** view, select the ‘+’ button to add a monitor

Choose monitor type and connection; create a new connection if necessary

Double click new monitor to start
**Exercise**

1. Start with your ‘shallow’ project
2. Create a run configuration
3. Complete the Resources tab
4. Select the executable in the Application tab
5. Submit the job
6. Check the job is visible in the Inactive Jobs view, moves to the Active Jobs view when it starts running (although it may be too quick to show up there), then moves back to the Inactive Jobs view when completed
7. View the job output
8. Remove the job from the Inactive Jobs view
Objective
- Learn how to use the HPC Toolkit to analyze the performance of your applications

Contents
- Introduction
- Application CPU Profiling using Xprof
- Profiling using hardware performance counters
- MPI profiling and tracing
What is HPC Toolkit?

- Set of performance tools integrated with Eclipse and PTP
  - Hardware performance counter profiling
  - MPI profiling and tracing
  - OpenMP profiling
  - I/O profiling and tracing
- Application CPU profiling
- Binary instrumentation tools
Hardware Performance Counters

- Helps identify problem code areas
  - Excessive CPU resource usage
- Collects hardware performance counter event counts
- Reports hardware performance counters for functions, blocks
- View multiple tasks
- Sort/filter data
IBM HPC Toolkit

MPI Profiling and Tracing

- Profiles MPI calls
  - Time spent
  - Number of calls
  - Message sizes
- Traces MPI calls
  - Traces individual MPI calls
  - Identifies communication patterns
  - Time spent/message size per MPI call
OpenMP Profiling

- Profiles OpenMP programs
  - Time spent in parallel region
  - Time waiting for barriers
  - Number of invocations of parallel region
  - Identify load imbalances across threads
IBM HPC Toolkit

I/O Profiling and Tracing

- Profiles I/O system calls
  - Time spent
  - Number of calls
  - Bytes per I/O
- Traces I/O calls
  - Traces individual I/O system calls
  - Statistics about I/O
Using HPC Toolkit
Possible workflow

- Compile – in preparation for profiling (-pg option)
- Run - to generate gmon.out files
- Run Xprof to determine ‘hot’ functions in application

- Instrument ‘hot’ functions in application for performance data gathering
  - Hardware Performance Counters, MPI, OpenMP, I/O
- Run the instrumented application
- Analyze application performance with HPC Toolkit tools

- Correct performance problems
  ... and repeat
Application CPU Profiling

- Profiles entire application
  - Shows time spent in each function
  - Easily identify highest CPU usage
  - Shows proportion of time in caller and callee
  - Additional views and reports for more detail
Compiling the Application
Compilation Requirements

- Compile flags for Hardware performance counters, MPI, OpenMP and I/O profiling
  - AIX: --g
  - Linux: --emit-stub-syms -Wl,--hash-style=sysv -g

- Compile flags for CPU profiling with Xprof
  - AIX and Linux: Application must be compiled and linked with -g and -pg flags
Xprof (CPU Profiling)
Xprof and Remote Systems

- Xprof not yet integrated with Eclipse
- Useful tool for getting overview of application performance
- Xprof must be run on remote AIX or Linux system
- Application can be run on remote system
- Application can also be run using PTP
- Tutorial fragment explains how to run on remote system
Set Up Makefile (Xprof)

- Makefile should look like following view (shows AIX)
- Should have **all** and **clean** targets
Build the Application

- Make source code changes and save
- Right click project name in **Project Explorer** view
- Click **Build Project** in popup menu
  - Or select project and use hammer icon in toolbar
- Force build by clicking **Clean Project** then **Build Project** in popup menu
  - Or Menu: Project ➤ Clean….
Build the Application 2

- Project Explorer updates to show new files
- Console view shows build log
Log on Login Node

- Open xterm on local machine
- ssh w/X11 tunneling to login node (2 hops)
- ssh tunneling for X11 must be enabled (by admins)
Set Up and Run Application

- Set PE environment variables
- Run application

```bash
export MP_PROCS=5
export MP_TASKS_PER_NODE=5
export MP_EUILIB=us
export MP_RESID=yes
export MP_RMPool=1
export MP_EUIDevice=sn_single
export MP_DEVTYPE=hf
$ cd ~/shallow
$ ./shallow
```
Display results using Xprof

- Set up HPC Toolkit environment (env.sh)
- Invoke Xprof with executable and gmon.out files
- Can specify multiple gmon.out files
Xprof Initial View

- Large green box for executable and each shared library (AIX)
- Nodes represent functions, lines represent caller/callee relationship
- Size of block representative of time in function
  - Width: Function & all of its callees
  - Height: Function only
Expand Application Tree

- Right click on outer green box
- Click **Remove Cluster Box** in popup menu
- Left click on nodes (boxes) to see function detail
- Left click on arcs to see caller/callee details
- Right click node or arc for additional actions
- Use **View** menu to zoom
- Use scrollbars to scroll
Filter Call Graph

- Click on **Filter** menu
- Click on **Filter by CPU Time**
- Drag slider to desired number of functions
- Lowest time functions filtered out
- Additional filters
  - Call counts
  - Function name
- Click **Show Entire Call Tree** to restore
Call Graph Profile Report

- Click on **Report** menu
- Click on **Call Graph Profile**
- Report shows
  - Time spent in function and descendants
  - Number of times a function was called

---

**HPCT-22**

**HPC Toolkit**
Hardware Performance Counter Profiling
Rebuild Application

- Update Makefile to use correct compile/link flags (shows Linux)
  - CFLAGS = -g -pg -q64
  - HFLAGS = -Wl,--emit-stub-syms -Wl,--hash-style=sysv
- Rebuild application using Project->Clean Project then Project->Build Project
Set up Profile Configuration

- Select **Run** in main menu
- Select **Profile Configurations** in menu
- Assuming you already have a configuration for ‘shallow’ ...
- Select existing run configuration from **Parallel Application** tab
- Make changes on following pages as needed
Set up Profile Configuration 2

- Launch config information from previous run
Set up Profile Configuration 3

- Switch to **Performance Analysis** tab
- If not already set, **Select Tool: HPC Toolkit**

Wait while the executable is analyzed “Analyzing Binary”
Set up Profile Configuration 4

- Select the **Common** tab
- Check the **Generate unique filenames** checkbox
  - Adds job PID to performance output filenames
Profile Configuration: HPM 1

- Select **HPM** tab
- Check what points to instrument
  - By function
  - By callsite
- Be selective to reduce overhead
- Example: select
  - Function body
  - calc.c
  - time.c
  - tstep.c

Scroll to select more...
Profile Configuration: HPM 2

- Select **Derived Metric Name** (e.g. MIPS)
- Select **Generate visualization files**
- Select **Profile button** to run the profile analysis
- Program is instrumented then executed
View HPM Performance Data

- Popup appears after application runs, click Yes to open viz files
- HPCT Perspective with Performance Data view opens
- Make sure performance data files are selected
  - Can select files with click, shift-click, ctrl-click
  - Deselect task 0 to get compute data info
Open the **Performance Data Summary** view

- When viewing multiple tasks
  - Right click on any entry in **Performance Data Summary** view
  - Select **Show Data for Other Rank**
  - Select task/pid from dropdown and click **OK**
View HPM Performance Data 3

- To view additional metrics
  - Select instrumentation point in **Performance Data Summary** view
  - Right click instrumentation point
  - Select **Show Metric Browser** view
- The **Metric Browser** view opens
  - Shows data for all tasks
View HPM Performance Data 4

- Pick sort column by selecting column title
- To add or delete visible metrics
  - Right click in **Metric Browser** view
  - Select **Show Other Metrics**
  - Hide or show metrics using **Filter Dialog**
To show all data for all tasks in a table:
- Right click in Performance Data Summary view
- Select Show Data as a Flat Table
- Performance Data Detail view opens
- Sort data by clicking desired table column
Re-opening Performance Data

- To re-open Performance Data files:
- Select **Project Explorer** view
- Select one or more data (.viz) files
- Right click over selected files
- Click **HPCT** in popup menu
- Click **Open performance files** in second menu
MPI Profiling and Tracing
Lab Exercise Note

- For the hardware performance counter exercise m and n = 512 in decs.h to get useful event counts and times
- Change m and n to 8 in decs.h and rebuild shallow to reduce trace size to a manageable level
  - m,n = 32 is ok
Collect MPI Data

- Open **Profile Configuration** dialog
- Example: Reuse profile configuration created for hardware performance counters
  - To remove HPM info: HPM tab, uncheck Function Body & reset DRM "MIPS" to blank and reset ‘generate vis. Files’
  - Or can copy profile configuration to keep both configs
- Same setup as hardware performance counters except you select **MPI** in **Performance Analysis** tab of profile configuration
Collect MPI Data (2)

- Select **MPI** tab
- Select instrumentation points
  - By call site
- Select checkbox to instrument this point
  - Be selective to reduce overhead
- For example:
  - Within `neighbour_receive`
  - Select `MPI_Recv_##`

**HPC Toolkit**

HPCT-40
Collect MPI Data 3

- Check **Output trace for all tasks**
- Check **Trace all MPI calls**

Select **Profile** to run the profile analysis
View MPI Profile Data

- **Performance Data Summary** view opens automatically
- Works similarly to viewing hardware performance counter data
  - Right click in view to open metric browser, show flat table, view different task’s data
- Right click somewhere in **Performance Data Summary** view
- Select **Load MPI Trace** in popup menu
- Select path to save trace file on *local* system

*HPC Toolkit*
View MPI Trace Data

- Timeline view of MPI function calls
- MPI Function calls color coded matching list of right
- Exclude functions from trace by clicking checkboxes
- Zoom in and out using buttons in title bar
- Right click on an event box to see details for that event
In Summary

- Set of performance tools integrated with Eclipse and PTP
  - Hardware performance counter profiling
  - MPI profiling and tracing
  - OpenMP profiling
  - I/O profiling and tracing
- Application CPU time profiling
- Binary instrumentation tools
IBM Parallel Debugger

Objectives
- Introduce the IBM Parallel Debugger
- Learn how to use the debugger features

Contents
- Overview of debugger architecture and features
- Lab #1: Scavenger Hunt
- Lab #2: Shallow
- Lab #3: Sample Sort
DARPA Challenge

- IBM’s PERCS
  - Productivity, ease-of-use
  - Debug applications executing at Petascale
  - Conventional 'serial' debug overwhelmed

“Debugging code on a parallel machine with hundreds or thousands of cores creates unique problems, and may be the biggest single challenge facing parallel programming”
  - Charles Holland [DARPA], The Economist, June 2011
About IBM ParDeb

- Utilizes mature serial debug engine technology
  - idebug +15 years in development
- Parallel Group Debug Infrastructure (PGDI)
  - Tree topology for efficient aggregation/filtering
- Customized for programming models
  - MPI, UPC, X10, openMP
- Efficient use of UI resources (pixels, mem, etc.)
- Emphasis on grouping
Parallel Debugger Architecture
IBM Parallel Debugger

Debugger Execution Environment

1. Client launches debug session
2. Proxy launches application and root debug agent
3. Root agent deploys PGDI
4. One leaf agent per OS image attaches to application processes

PGDI = Parallel Group Debug Infrastructure
Debugging at Petascale (1/2)

- 1,000,000 (1M) execution threads
  - How can they be represented in the UI?
  - How can the user operate on them?
    - Need to be able to operate on a subset of threads
  - How can the user find the defect?
    - Not practical to examine each thread one by one
Debugging at Petascale (2/2)

- Organizing the threads into groups
  - Based on common characteristics of thread state and data
  - Can’t be done manually
  - What about thread state and data changes?
  - This only works if the debugger does all the heavy lifting
Debug Groups

- Debug groups are dynamic collections of user threads.
- All threads belong to at least one group.
- Threads join and leave groups as session progresses.

Group types:

- ALL – all user threads
- Debug state – application states (e.g. SUSPENDED)
- Function location – suspended in a function or method
- Line location – suspended on a line in a source file
- Breakpoint – suspended on a breakpoint
- Stack – threads that share a similar call stack
- Expression – threads with common data property
- Distribution – threads with a variable within a certain range
- Static – copy of some other group, membership never changes
Scalable Group Synthesis (1/4)

100 threads to manage

2 groups managed manually

2 synthesized groups (automatically classified)

Think about 1M threads!!!

Debug-8
Scalable Group Synthesis (2/4)

2 synthesized groups
(automatically classified)

2 items
Scalable Group Synthesis (3/4)

- Groups are synthesized by applying a set of conditions to an existing group
  - Group_{\text{ALL}} and Suspended $\Rightarrow$ Group_{\text{Suspended}}
  - Group_{\text{ALL}} and Running $\Rightarrow$ Group_{\text{Running}}
- Members join/leave groups dynamically
- Membership is not enumerated
  - PGDI encodes membership during synthesis
  - Group representative available to UI
- Billions of groups can be reserved
  - Light weight
  - Group vectors/arrays
Scalable Group Synthesis (4/4)

- Problem investigation strategies
  - Divide and conquer
  - Odd man out
  - Outliers
  - Etc...
  - Can be applied in combination
Debug Perspective
IBM Parallel Debugger

Debug View

- Used to control groups of threads
- Debug actions operate on threads associated with selected group
- Group name indicates type
- Number of threads in group is shown in [ ]’s
- Focused member section shows stack frame of representative from selected group
- Group selection drives
  - Source View
  - Parallel Stacks View
  - Variables View
Source/Editor View

- Synchronized with debug view
- Left Margin
  - Breakpoint markers
  - Execution context markers
- Right Margin
  - File level execution markers
- Line highlighting for suspended threads
  - Orange shading
    - Active thread(s) selected in Debug view
  - Blue dashed box
    - Active thread(s) not selected in Debug view
  - Green shading
    - Thread(s) selected in Focused Member section
Parallel Stacks View

- Shows merged call stacks for the group selected in the Debug view
- Works with debug control actions to step or resume threads
- Toggle function level or line level comparison
- Toggle to select call hierarchy order
- Stack selection drives
  - Source View
  - Group Details View
Variables View

- Displays variables from representative member of selected group
- Shows variable value and type information (if available)
- Yellow highlight when value is modified during execution
- Value can be edited, changes will be reflected when execution continues
- Data structures are represented using hierarchical tree format
- Variable values can be auto-refreshed at regular intervals
Parallel Debug Groups View

- Acts as a clipboard area
- Interesting groups can be saved and monitored during a debug session
- Right-click on group in debug view and select **Save Group**
  - Will place group in Parallel Debug Groups view
Data Elements View

- Used to retrieve a number of specific data values from a large set of data elements
- Data elements can be
  - Scalar local variables
  - Scalar elements of shared or local array
- Elements are retrieved from all suspended threads that have the variable in scope
  - Value matches value range
- Application must be built with FD2 Debug Extensions
Data Distributions View

- Histogram of data element values
- Each bar corresponds to value sub-range
  - Length of bar represents number of elements in range
- Upper and lower bounds can be calculated automatically or specified manually
- Distributions can be created on
  - Scalar local variables
  - Scalar elements of a shared or local array
- Requires application built with FD2 Debug Extensions
Parallel Group Debug Configuration
Lab #1 – Scavenger Hunt

- Parallel debug perspective
- Debug view
  - Groups (ALL, Suspended, Function, Line)
  - Focused Member
    - Group representative's call stack
  - Debug controls (resume, step into, step over, etc...)
  - Breakpoints
  - Selection drives content of other views such as
    - Source view, Variables view, Parallel stacks view, etc...
Lab #2 - Shallow

- Debug an MPI application
  - MPI Debug Library
  - MPI Error Handling
- Reinforce basic debug skill set
  - Breakpoints
  - Stepping (step into, step over, step return)
- Debug, code, build cycle
Lab #3 – Sample Sort

- Debug a UPC application
  - View shared variables
  - Use of UPC Barrierpoints
- Diagnose errors
  - Program doesn’t run to completion
  - Determine cause and resolve through basic and advanced debugging features
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
Useful Eclipse Tools

- Linux Tools (autotools, valgrind, Oprofile, Gprof)
  - http://eclipse.org/linuxtools
- Python
  - http://pydev.org
- Ruby
  - http://www.aptana.com/products/radrails
- Perl
  - http://www.epic-ide.org
- Git
  - http://www.eclipse.org/egit
- VI bindings
  - Vrapper (open source) - http://vrapper.sourceforge.net
  - viPlugin (commercial) - http://www.viplugin.com

Wrap-up
Online Information

- IBM PE Developer Edition
  - http://ibm.co/tdM7QD

- 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

Wrap-up
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 Monthly
  - Each second Tuesday, 1:00 pm ET
  - Details on the PTP wiki
- Attend the monthly user meetings
  - Teleconference Monthly
  - Each 4th Wednesday, 1:00 pm ET

Wrap-up
PTP Tutorial Feedback

- Please complete feedback form
- Your feedback is valuable!

Thanks for attending
We hope you found it useful