parallel tools platform

http://eclipse.org/ptp

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

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

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

Tutorial Outline

Time	Module	Topics
13:30 – 14:00	1. Overview of Eclipse and PTP	✦ Introduction to Eclipse/PTP ✦ Installation of Parallel Package
14:00 – 15:00	2. Eclipse Basics	✦ Importing a project ✦ Editing, building, launching ✦ Working with MPI, Fortran
15:00 – 15:30	Break	
15:30 – 16:00	3. Advanced Features	✦ Target system configurations ✦ Advanced development and refactoring
16:00 – 16:30	4. IBM HPC Toolkit	✦ Profile application using Xprof ✦ Hardware performance counters ✦ MPI Profiling and Tracing
16:30 – 16:55	5. IBM Parallel Debugger	✦ Overview of features ✦ Hands-on workshop exercises
16:55 – 17:00	6. Wrap-up	✦ Overview of features not covered ✦ Resources, getting questions answered ✦ How to participate ✦ Participant feedback

Installation

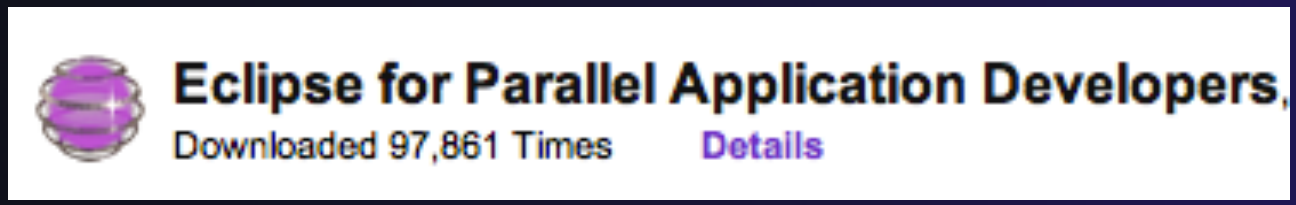
- Objective
  - To learn how to install Eclipse and PTP
- Contents
  - System Prerequisites
  - Download and Install Eclipse

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

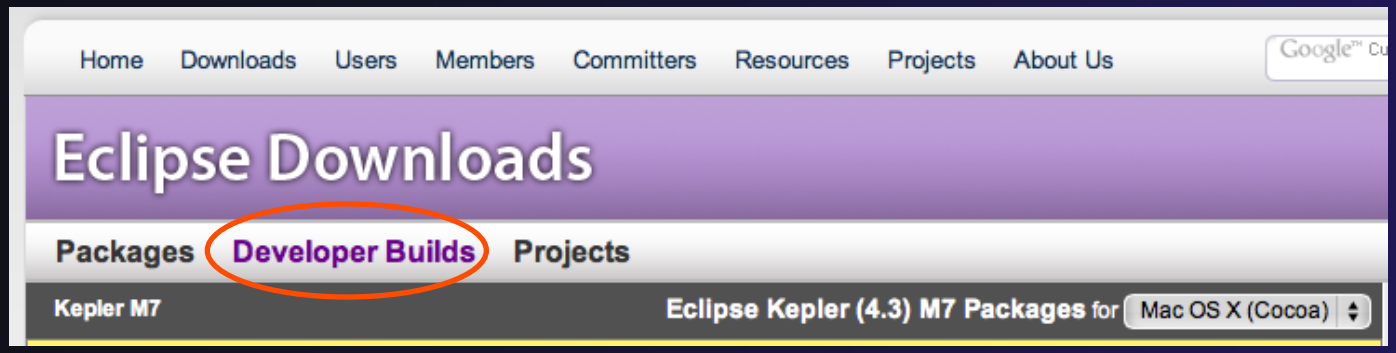
- The current version of Eclipse (4.2) is also known as "Juno"
  - Next release "Kepler" in June 2013
- Eclipse is available in a number of different packages for different kinds of development
  - http://eclipse.org/downloads
- For PTP, we recommend the all-in-one download:
  - Eclipse for Parallel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



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

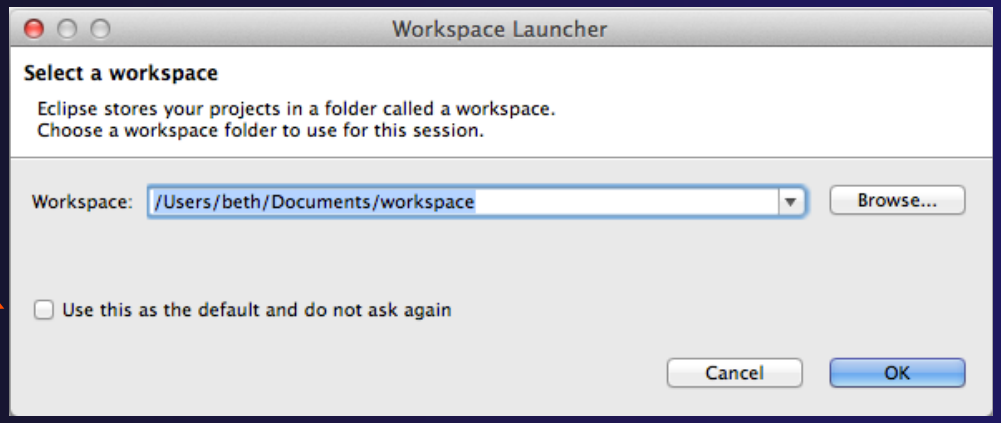
Starting Eclipse

- **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

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 default workspace location is fine for this tutorial

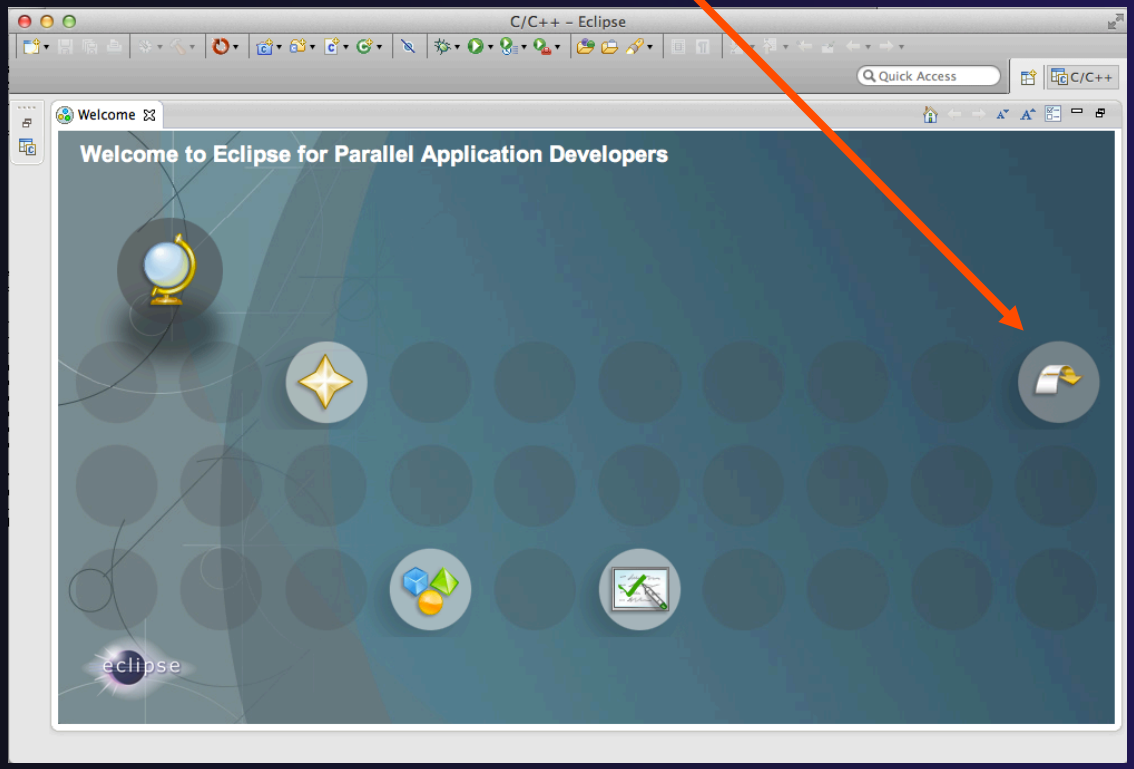
The prompt can be turned off



Eclipse Welcome Page

- Displayed when Eclipse is run for the first time

Select “Go to the workbench”

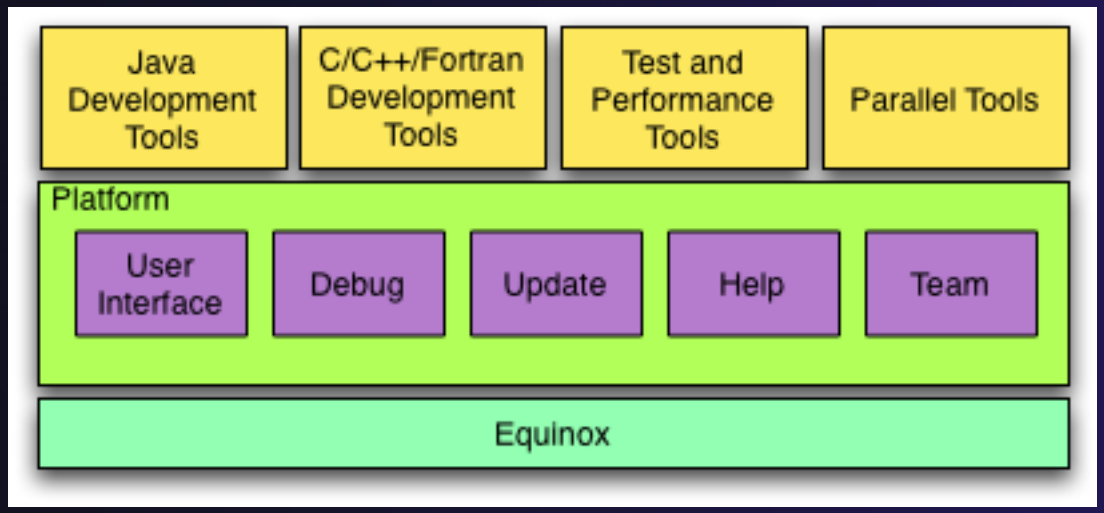


Introduction

- 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

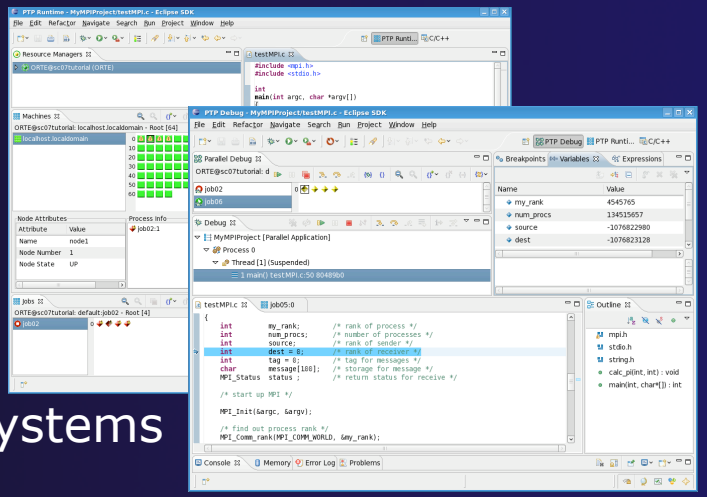


Eclipse Features

- 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

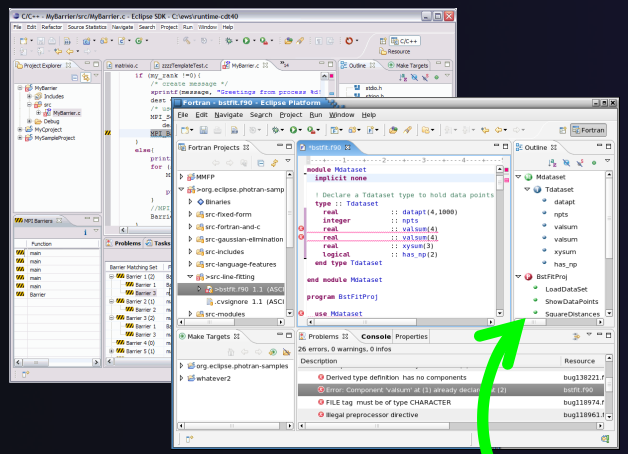
Parallel Tools Platform (PTP)

- 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, 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

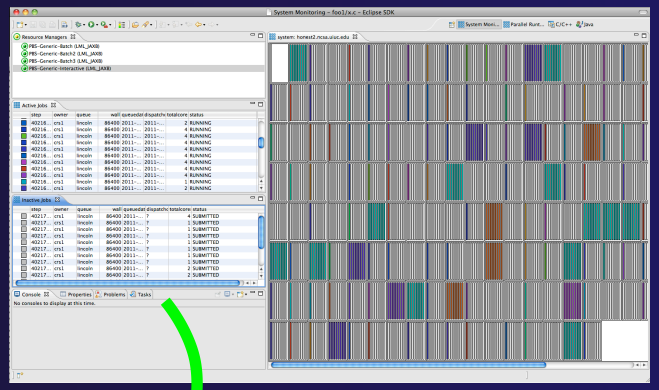


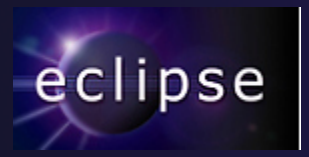
Eclipse PTP Family of Tools

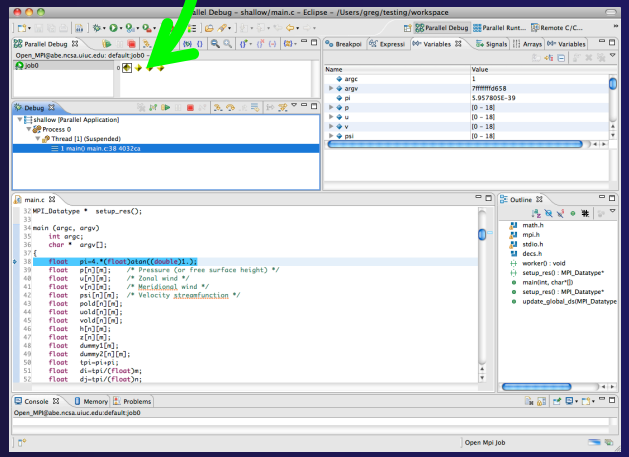
Coding & Analysis
(C, C++, Fortran)

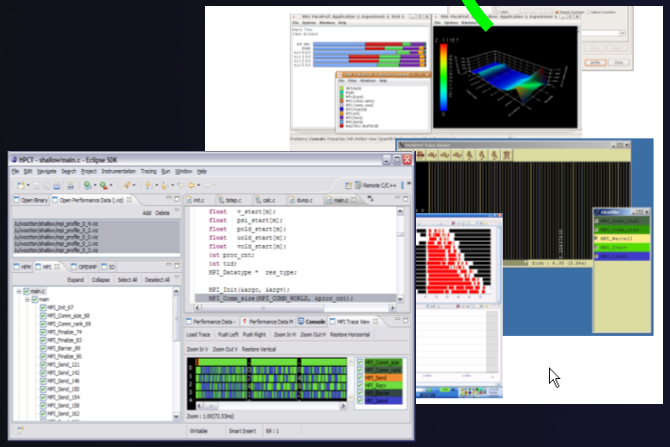


Launching &
Monitoring





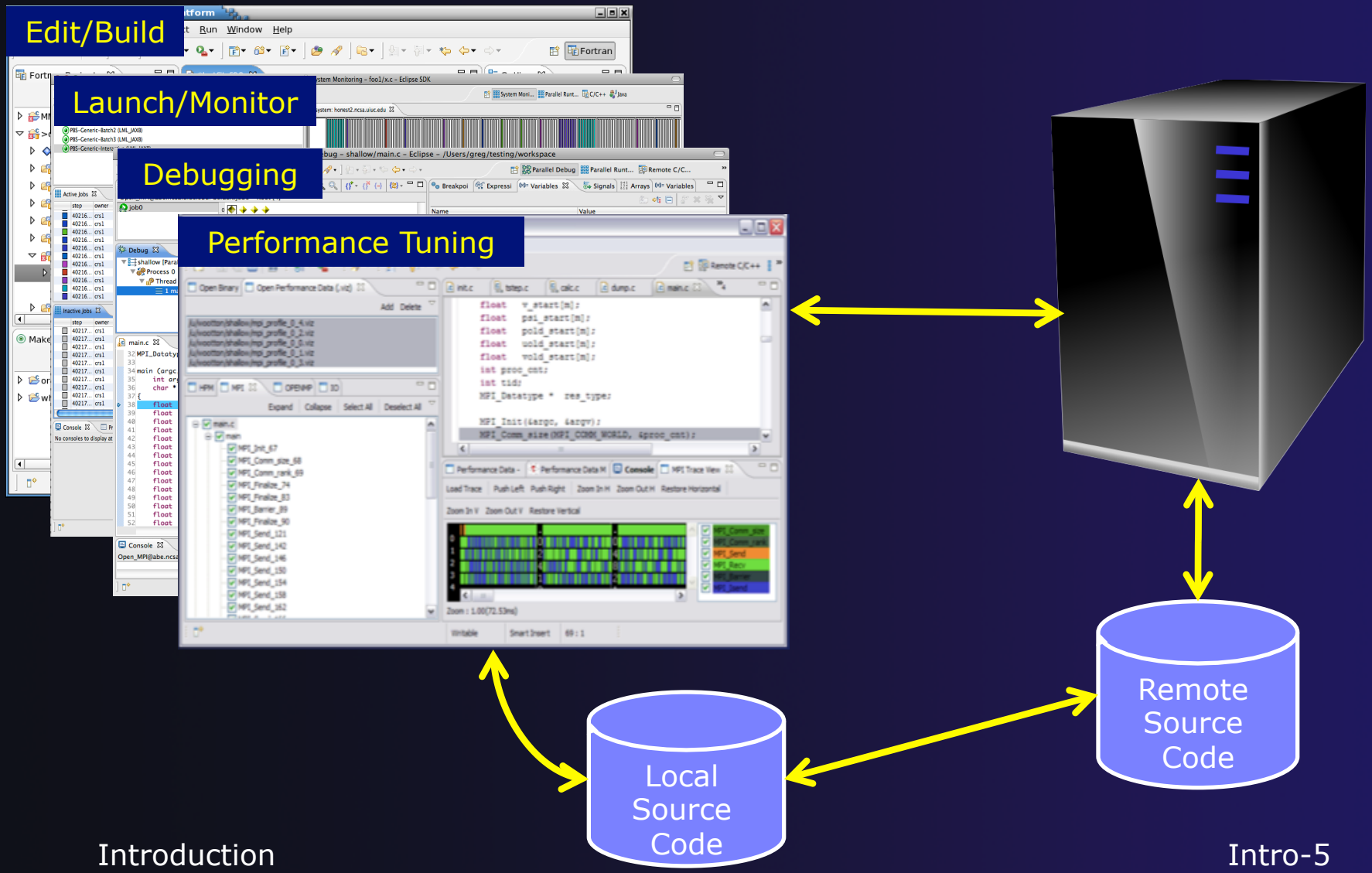




Parallel Debugging

Performance Tuning
(TAU, HPCT, …)

How Eclipse is Used



Parallel Environment Developer Edition 1.3

PE DE Client

Programming models	HPC Toolkit	Parallel Tools Platform	Base Tools Platform
• MPI • LAPI • PAMI • UPC • OpenMP • OpenSHMEM	• MPI profiling • OpenMP profiling* • Hardware counter profiling • I/O profiling • Profiling and trace visualization • Binary instrumentation* * Supported on Power only	• PTP 7.0 • Parallel Language Development Tools • Parallel application launch • System monitoring • Synchronized projects • Remote projects • External Tools Framework • LSF Support	Eclipse Platform (4.3) C/C++ Development Tools (CDT 8.2) Fortran Development Tools (Photran 8.1) Linux Tools

Windows | Linux | Mac OS X

PE DE Server

HPC Toolkit

Peekperf Visualization | Xprof Visualization | Instrumentation Libraries | Command-line Tools

PE RE

Scalable Communication Infrastructure (SCI)

POE Runtime

LoadLeveler | LSF

Power | x86 | BG/Q

Open MPI
MPICH2
PBS/Torque
SLURM
Grid Engine

Linux | AIX

http://ibm.co/12L2RxK

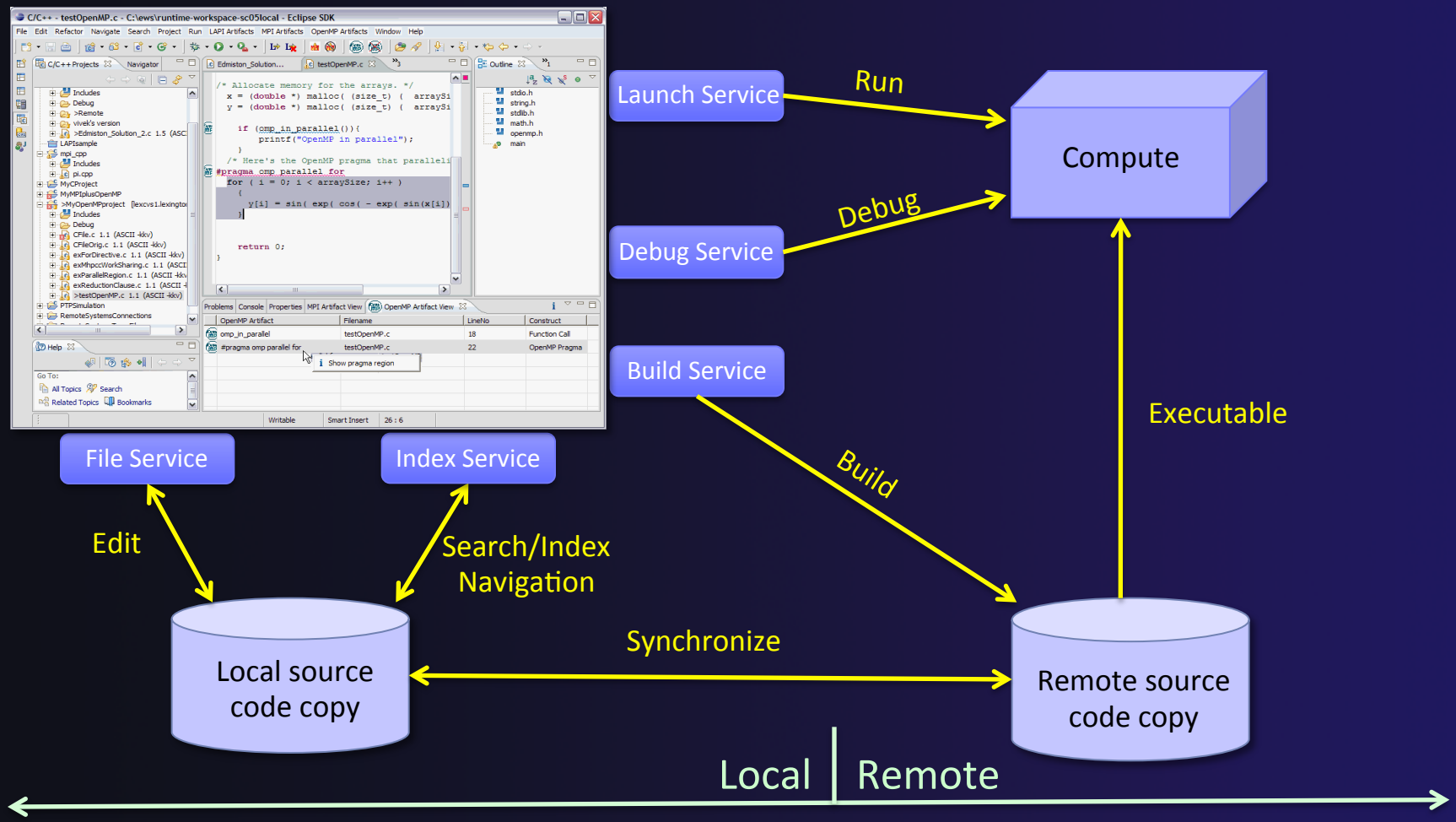
Importing a Project

- 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



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

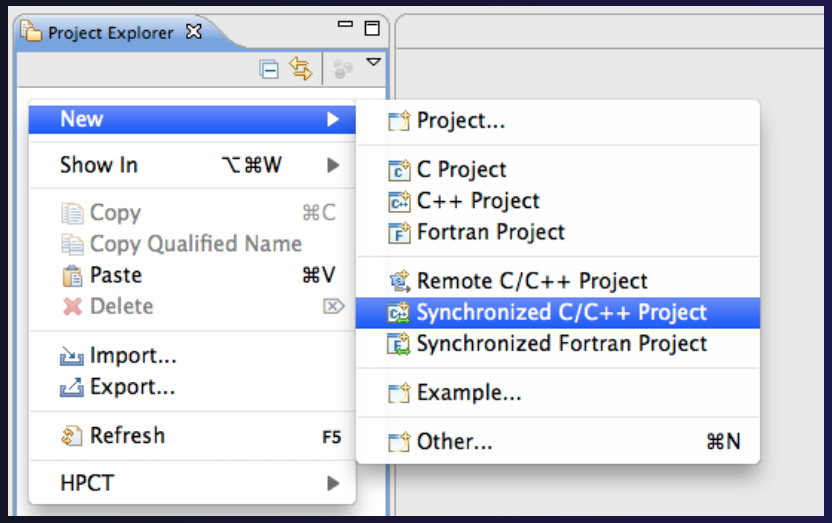
Source Code for project

- Source code exists on remote target

```
$ pwd
/gpfs/ibmu/tibbitts/shallow
$ ls -la
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

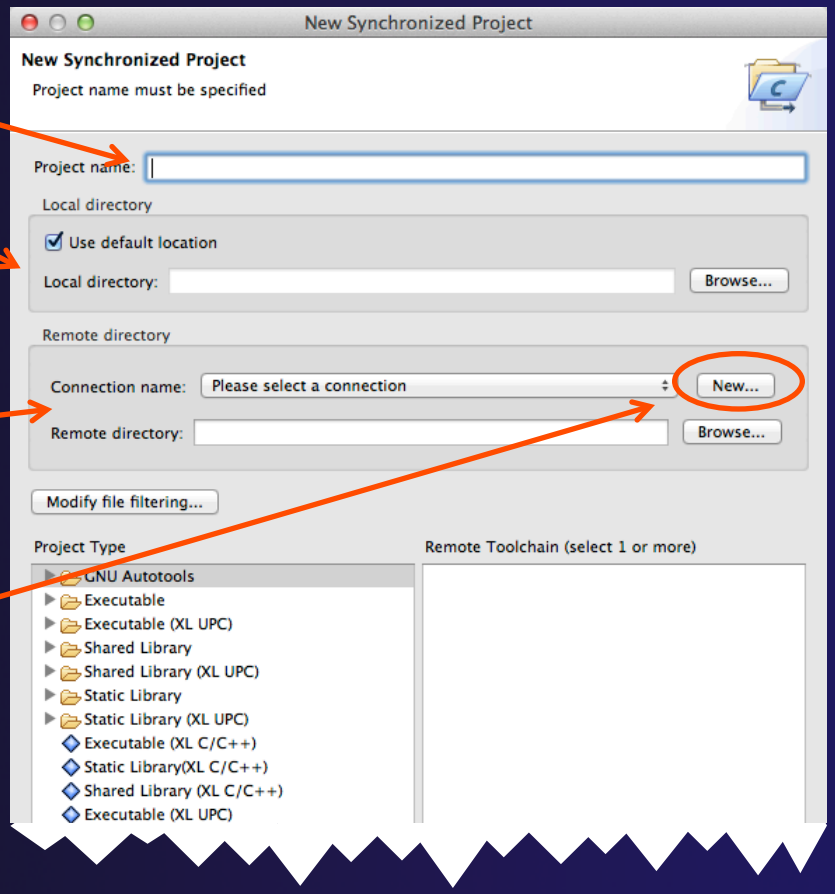
- In the Project Explorer, right click then choose
  - **New>Synchronized C/C++ Project** if your project is C/C++ only



  - **New>Synchronized Fortran Project** if your project contains Fortran files
  - This adds a Fortran nature so you can access Fortran properties, etc.

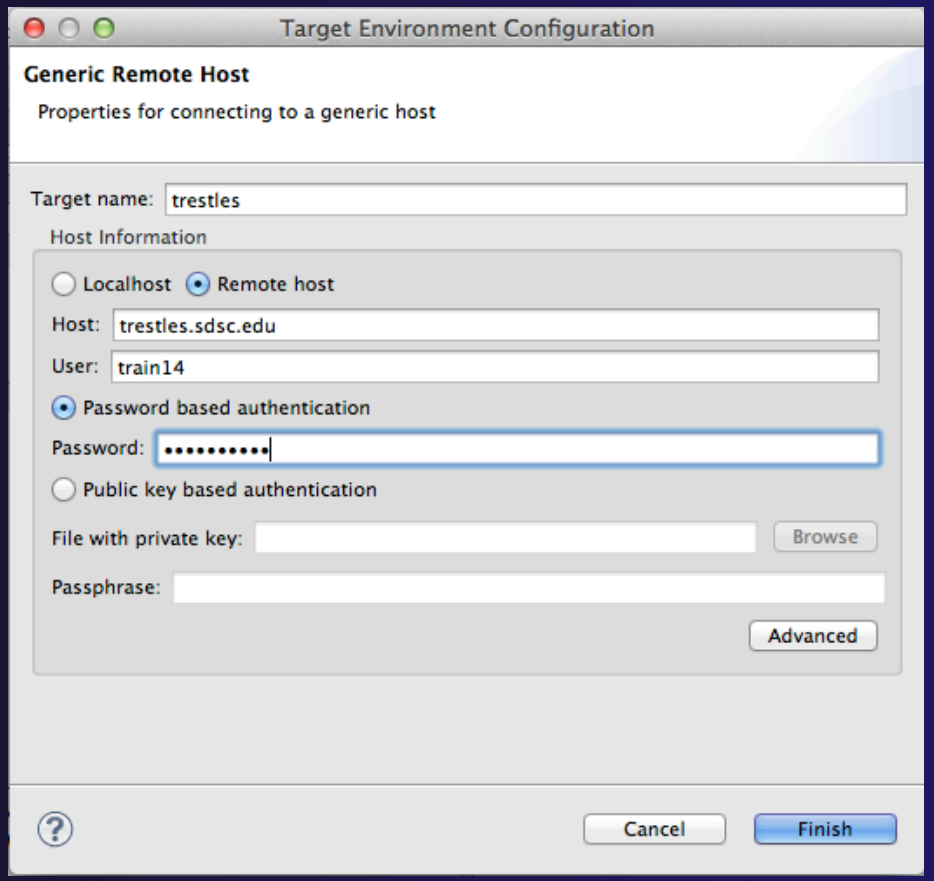
New Synchronized Project Wizard

- 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)



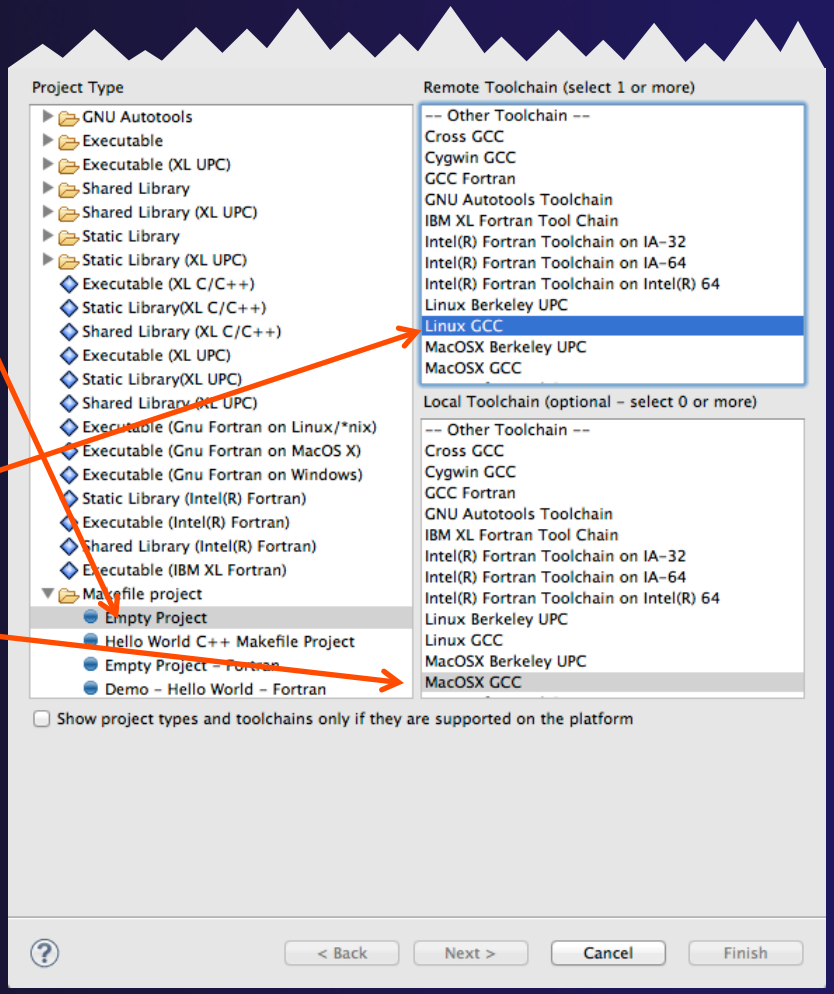
Creating a Connection

- 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**



Project Type & Toolchain

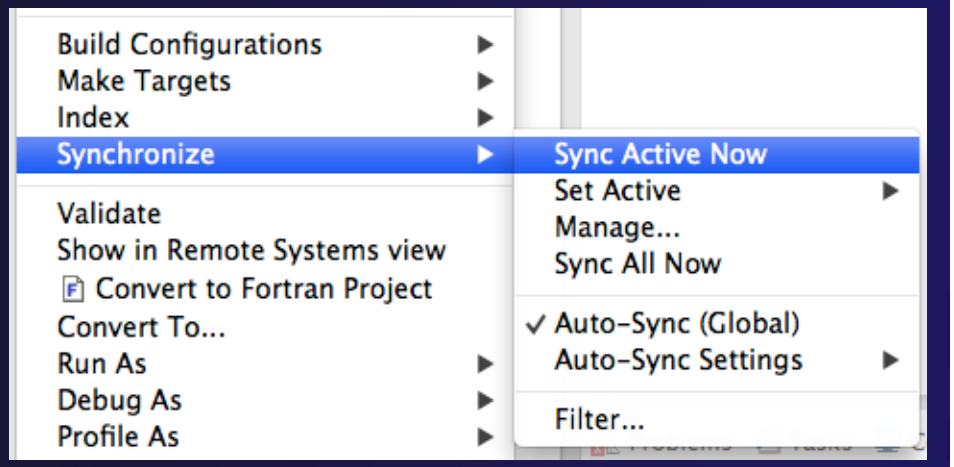
- Choose the **Project Type**
  - If you are synchronizing with an existing project, use **Makefile Project>Empty Project**
  - Otherwise, choose the type of project you want to create
- Choose the toolchain for the remote build
  - Use a toolchain that most closely matches the remote system
- Choose 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

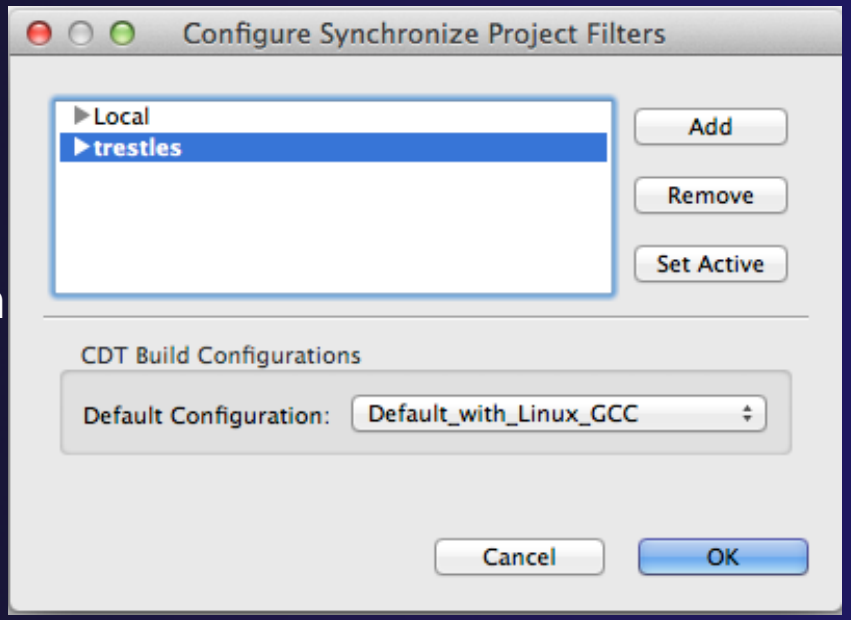
- Right click on project to access **Synchronize** menu
  - **Sync Active Now** will manually synchronize the active configuration
  - **Set Active** can be used to select the active configuration
  - **Manage...** is used to create new configurations to synchronize to different target systems
  - **Sync 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



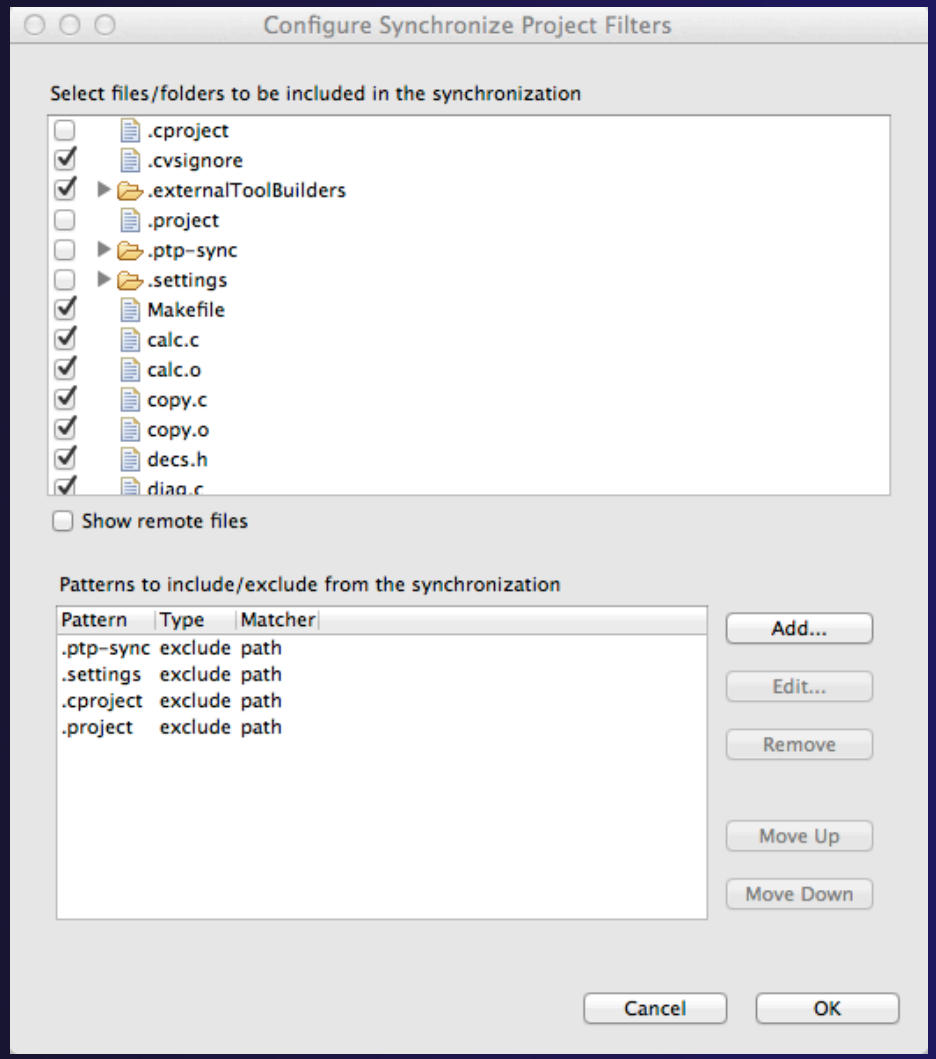
*By default, there will be a configuration for the target system (active) and a **Local** configuration. The Local configuration can be used to build a local copy of the project if desired.*

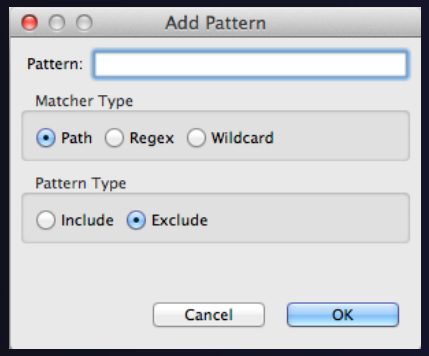
Synchronize Filters

- 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

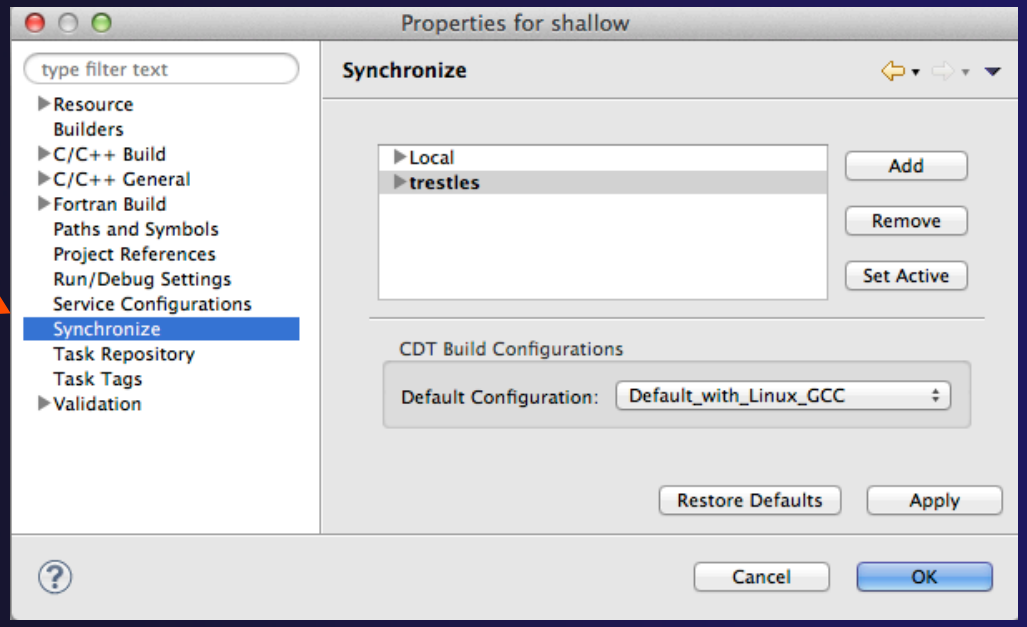
- Files can be filtered individually by selecting/unselecting them in the **File View**
- Include or exclude files based on
  - Paths
  - Regular expressions
  - Wildcards (foo.*)





Synchronized Project Properties

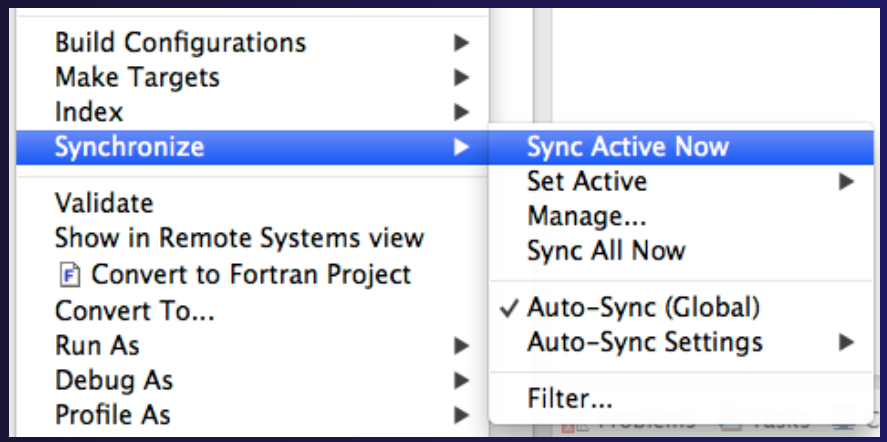
- Synchronized configurations can also be managed through the project properties
- Open the project properties by right-clicking on the project and selecting **Properties**
  - Select **Synchronize**
- This is the same as using the **Synchronize>Manage...** menu



Forcing a Resync

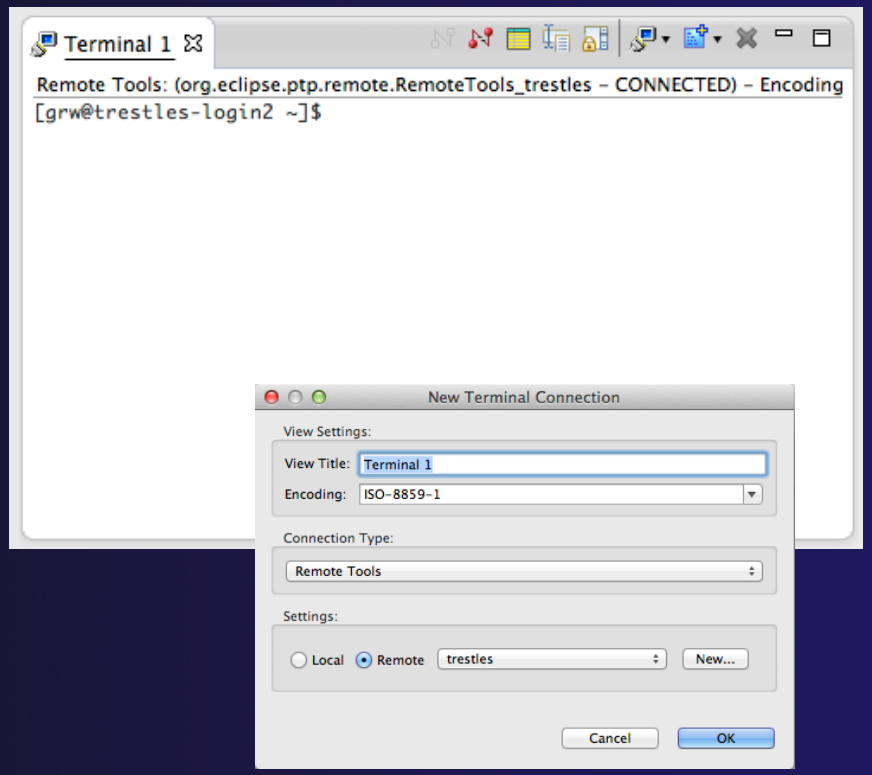
- If Auto-sync is set, the project should automatically resync with remote system when things change
- Sometimes you may need to do it explicitly
- Right click on project and select **Synchronization>Sync Active Now**

- Status area in lower right shows when Synchronization occurs



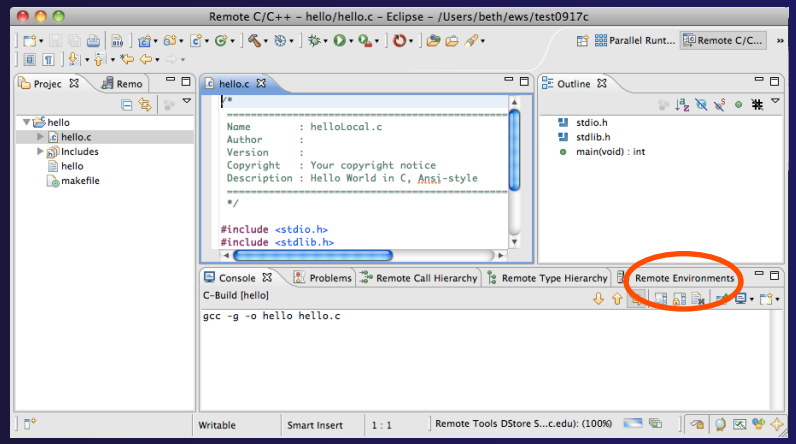
Remote Terminal

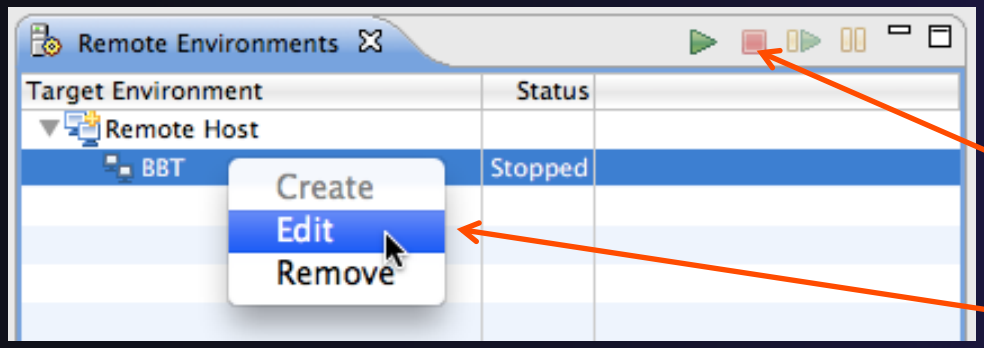
- There is a remote terminal that can be used to provide a shell from within Eclipse
- Select **Window>Show View>Other...**
- Choose **Terminal** from the Terminal folder
- In the **Terminal** view, click on the **New Terminal Connection** button
- Choose a previously configured connection from the dropdown,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

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

Optional Exercise

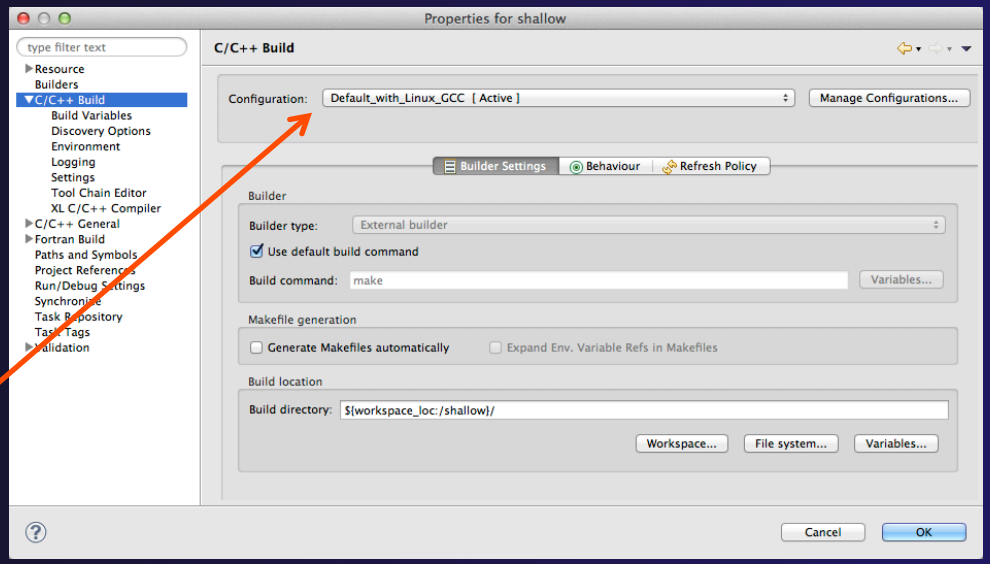
1. Experiment with sync filters
   - Create a filter to exclude files ending in ".o"
   - Create an empty file ending in ".o" on the local system and verify it is not copied to remote machine after a sync
   - Try creating a file on the remote system and verify that it is not copied
2. Create a new synchronize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

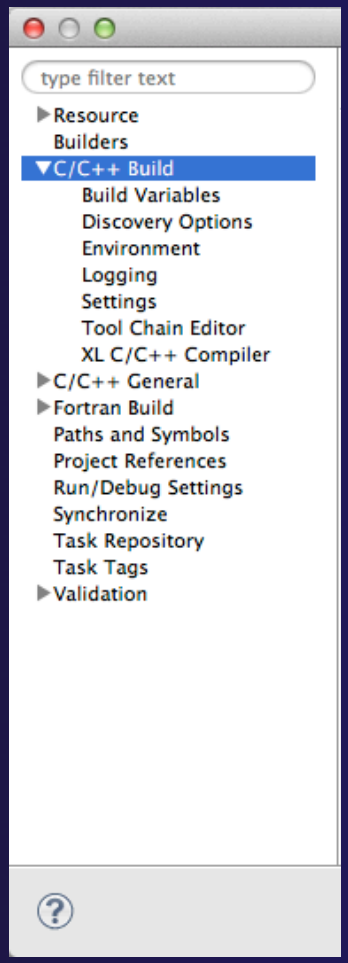
- A build configuration provides the necessary information to build the project
- The build configuration information is specified in the project properties
- Projects can have multiple build configurations, each configuration specifies a different set of options for a build
- Open the properties by 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

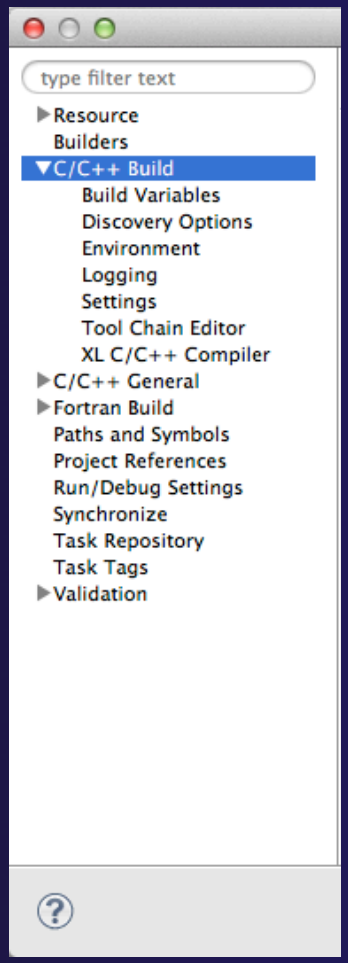
Build Properties (1)

- **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



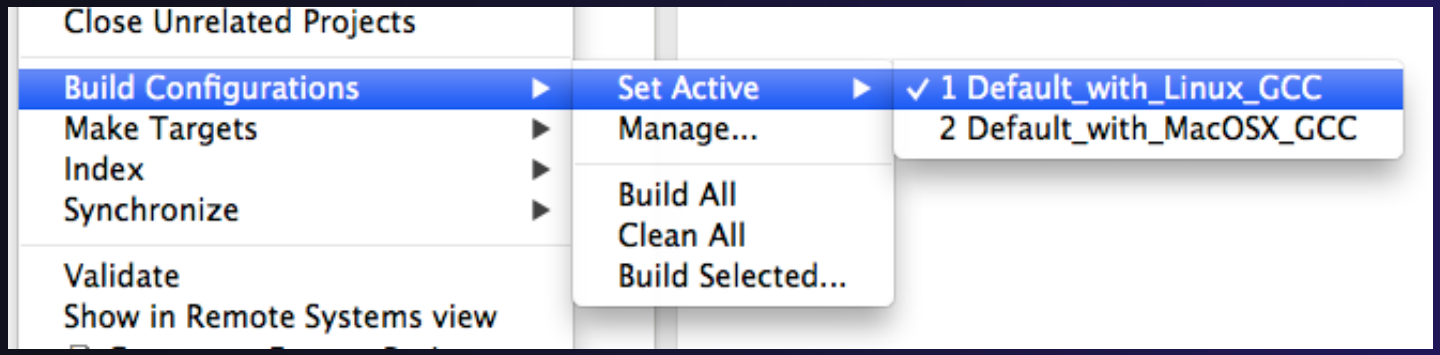
Build Properties (2)

- **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)



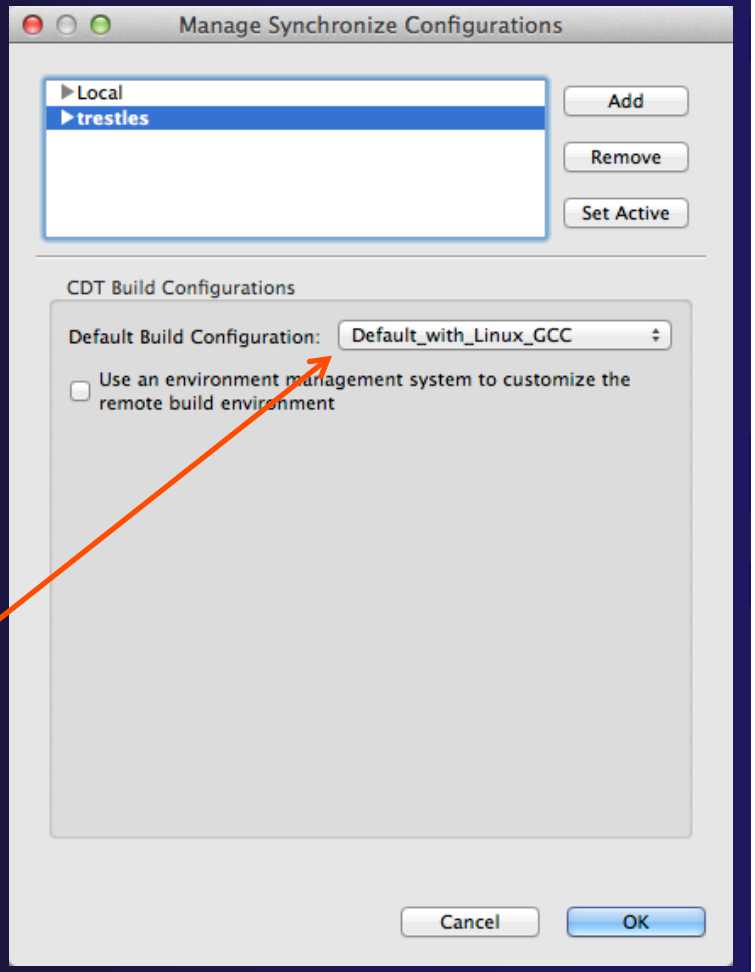
Selecting Build Configuration

- 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



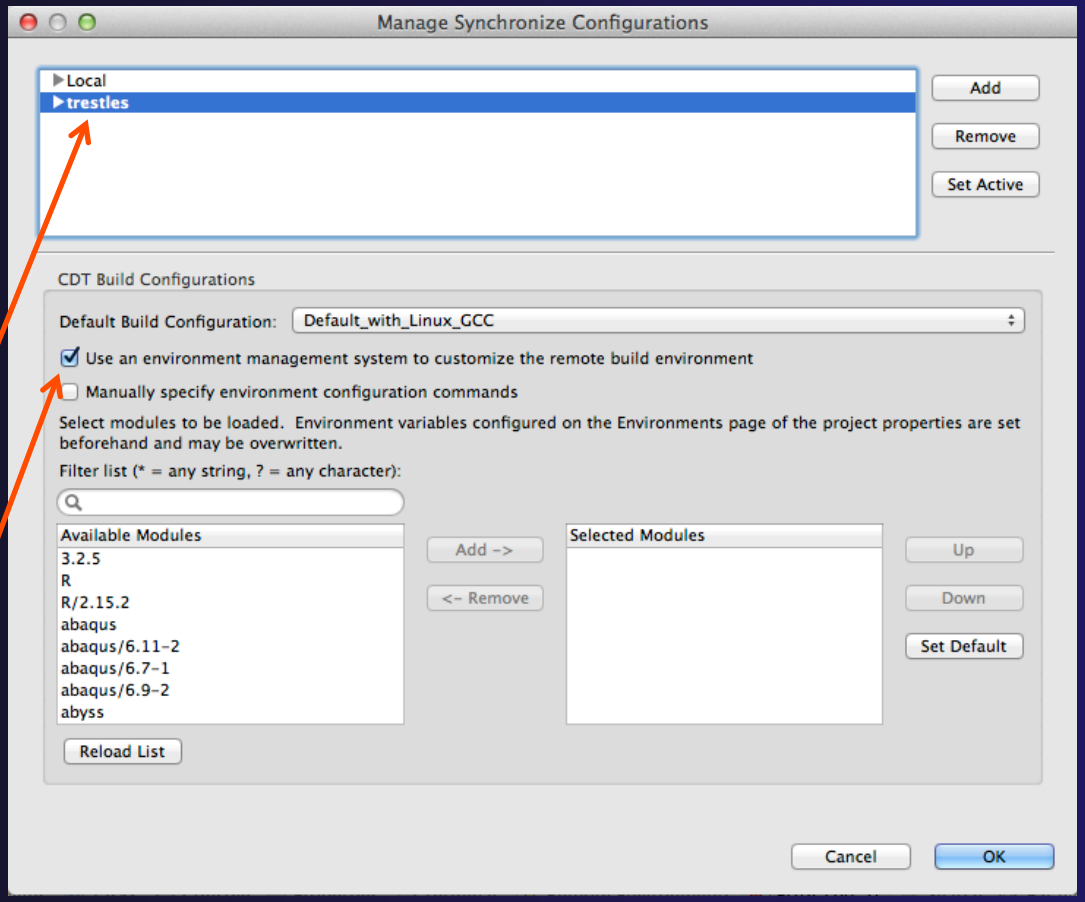
Building Synchronized Projects

- 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



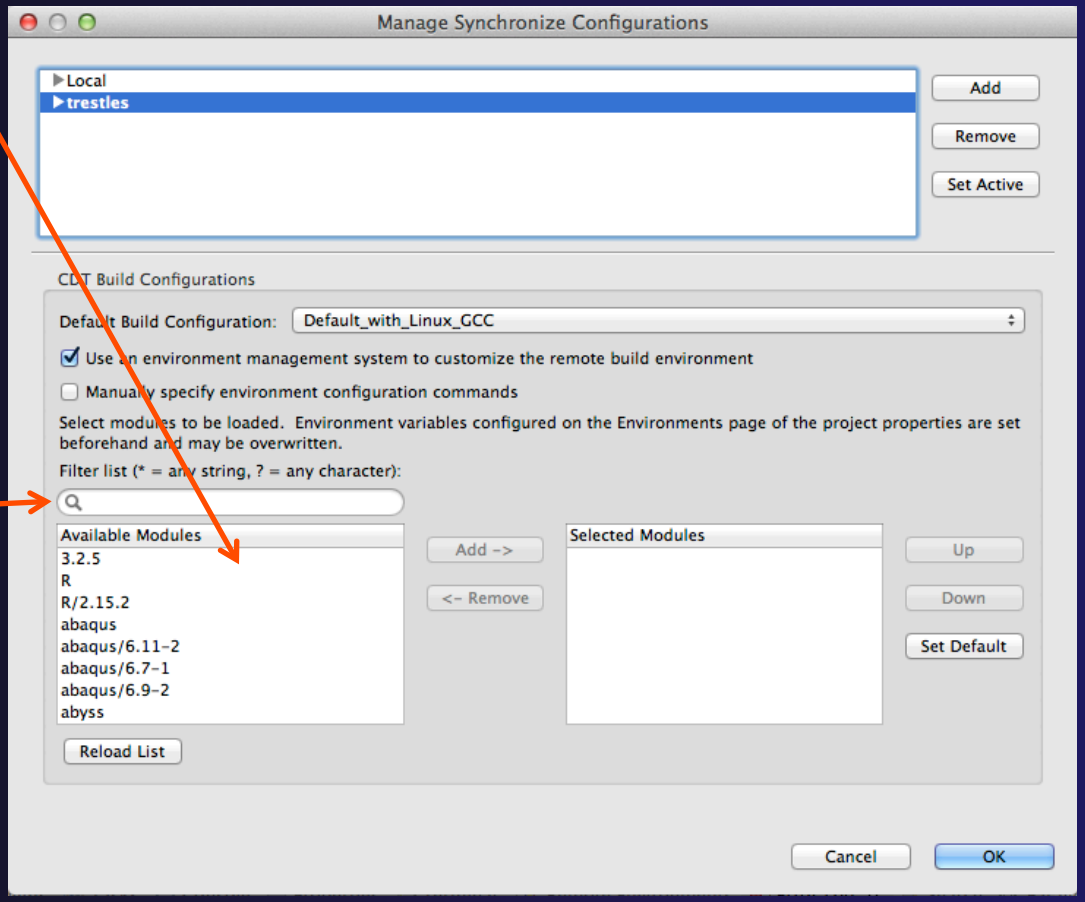
Configuring the Build Environment

- 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**



Build Environment (2)

- 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



Build Environment (3)

- 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

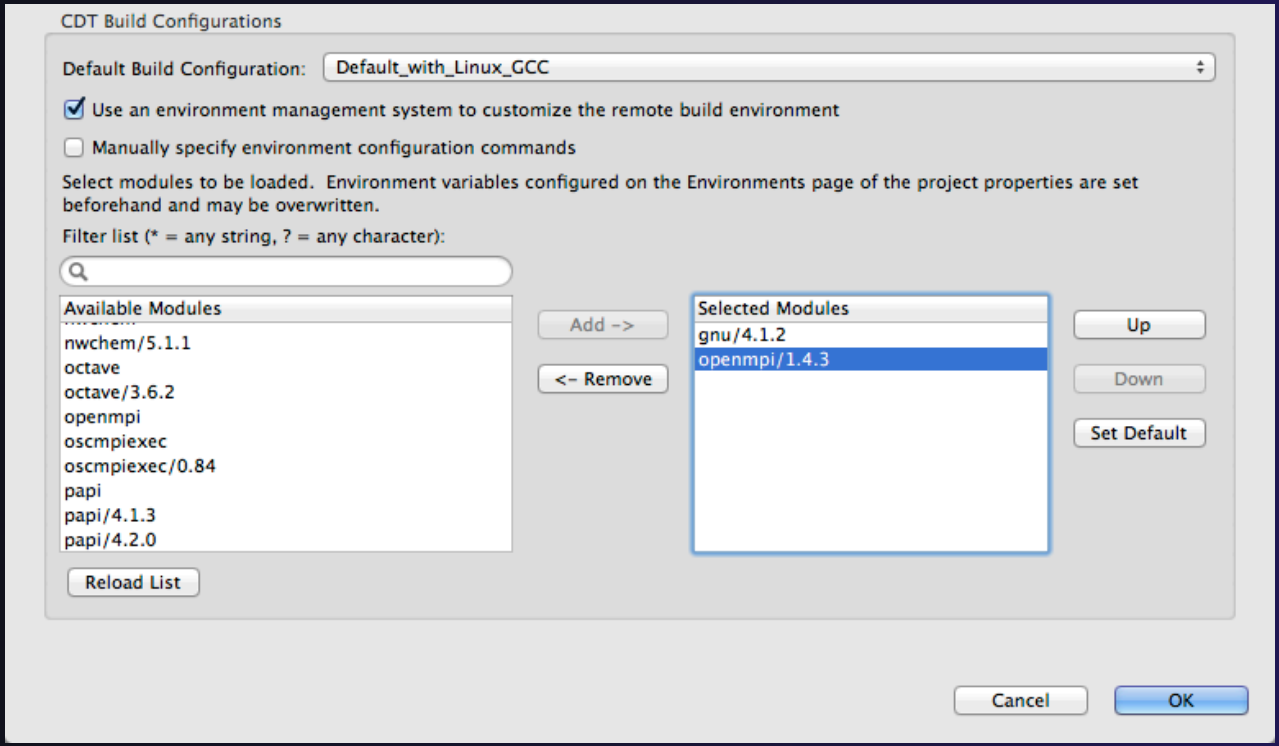
Console

CDT Build Console [shallow]

```
17:53:20 **** Build of configuration Default_remote for project shallow ****
make all

**** Environment configuration script temporarily stored in /tmp/ptpscript_rhMesG ****
module purge >/dev/null 2>&1
module load cuda-4.0.17
module load cupti/4.0.17
```

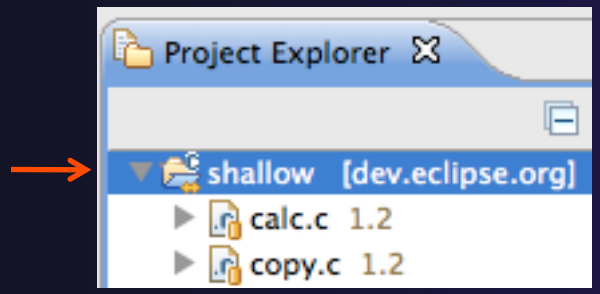
Build Environment (4)

- 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 ->**

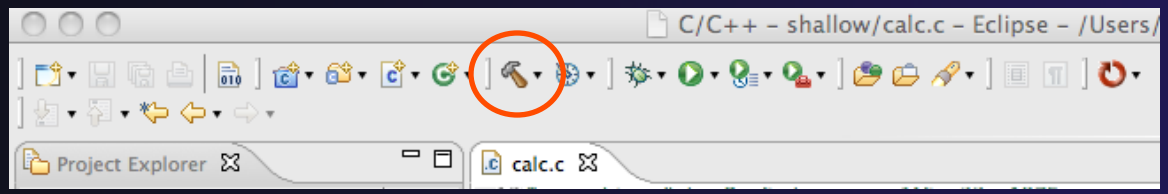


Starting the Build

- 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

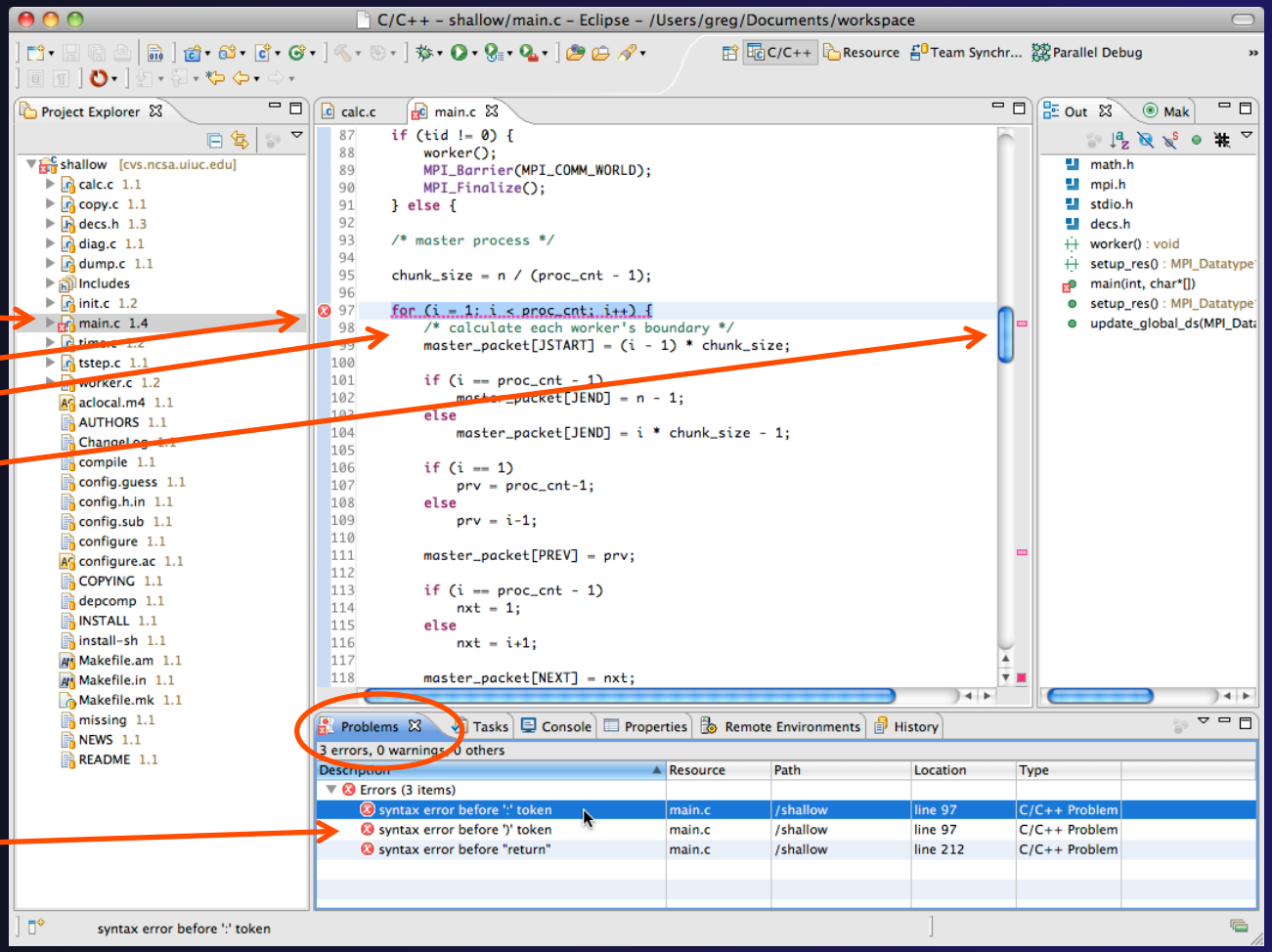
Problems | Tasks | Console | Properties | Remote Environments | Terminals

CDT Build Console [shallow]

```
09:06:10 **** Build of configuration Default_remote for project shallow ****
make all
Unloading compiler-dependent module openmpi/1.4.3
Note: mpicc appears to invoke gcc
mpicc -g   -c -o calc.o calc.c
mpicc -g   -c -o copy.o copy.c
mpicc -g   -c -o diag.o diag.c
mpicc -g   -c -o init.o init.c
mpicc -g   -c -o main.o main.c
mpicc -g   -c -o time.o time.c
mpif90 -g  -c -o tstep.o tstep.f90
mpicc -g   -c -o worker.o worker.c
mpicc -g   -c -o dump.o dump.c
mpicc -g -o shallow calc.o copy.o diag.o init.o main.o time.o tstep.o worker.o dump.o -lm -lgfortran

09:06:15 Build Finished (took 4s.578ms)
```

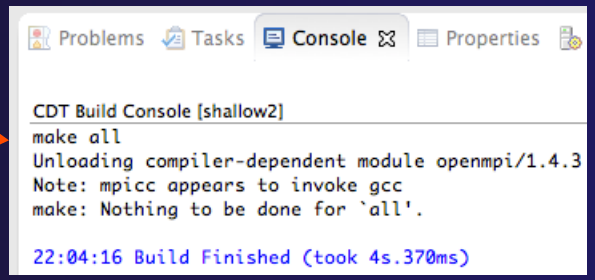
Build Problems

- 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

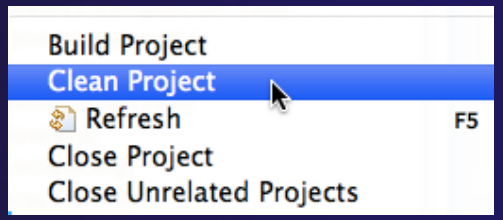


Forcing a Rebuild

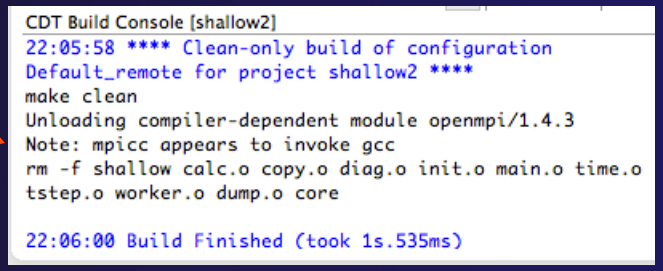
- If no changes have been made, `make` doesn't think a build is needed e.g. if you only change the Makefile



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



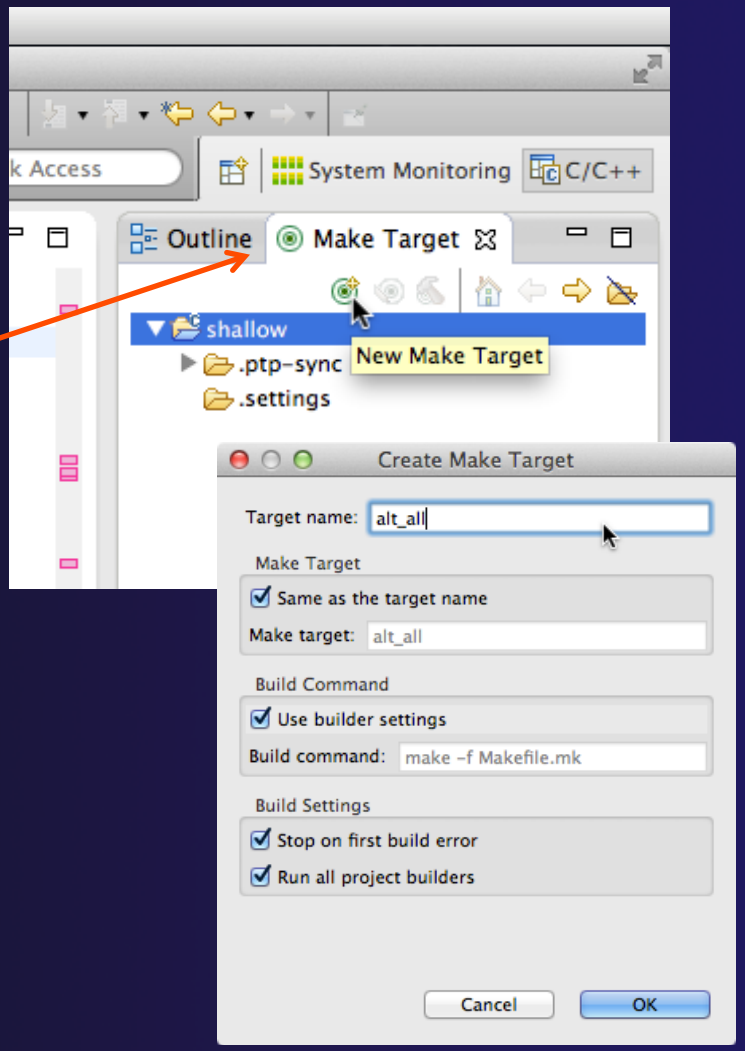
- Build console will display results of clean



- 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



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

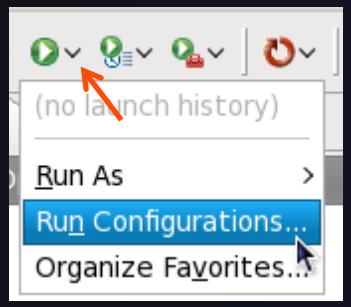
1. Open the Makefile in Eclipse. Note the line starting with "tags:" – this defines a make target named **tags**.
2. Open the Outline view while the Makefile is open. What icon is used to denote make targets in the Outline?
3. Right-click the **tags** entry in the Outline view. Add a Make Target for **tags**.
4. Open the Make Targets view, and build the **tags** target.

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

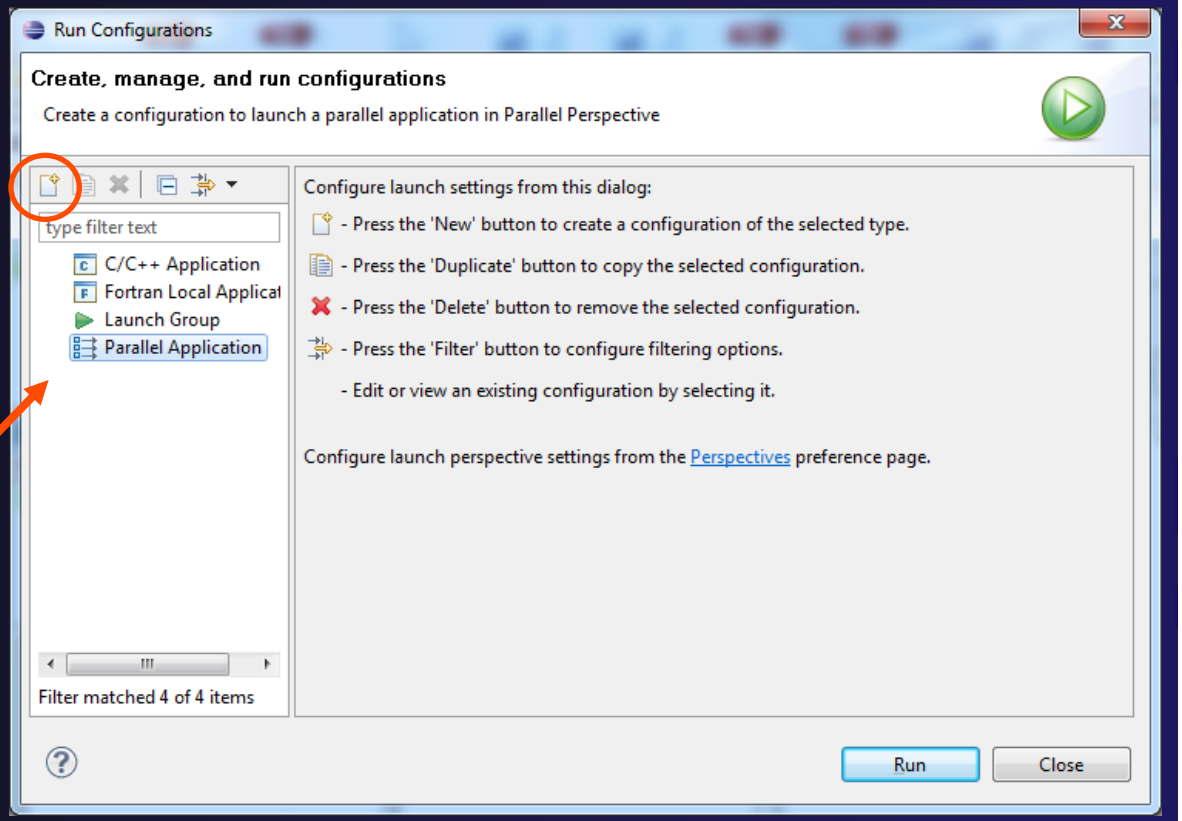
- Objective
  - Learn how to run an MPI program on a remote system
- Contents
  - Creating a run configuration
  - Configuring the application run
  - Monitoring the system and jobs
  - Controlling jobs
  - Obtaining job output

Creating a Run Configuration



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

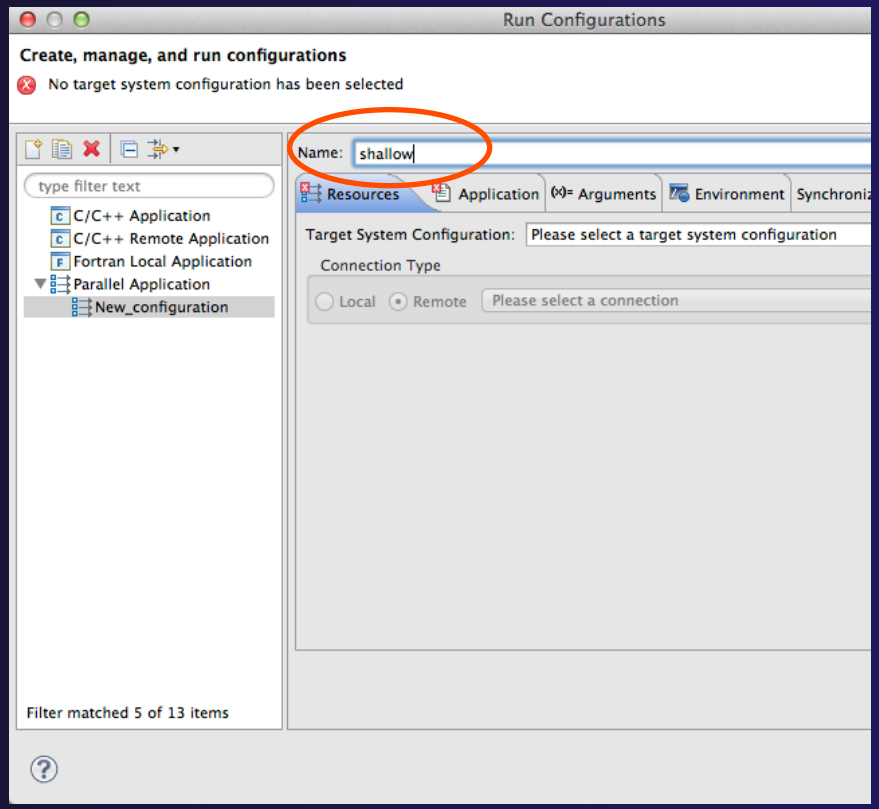
  Or, just double-click on **Parallel Application** to create a new one



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.

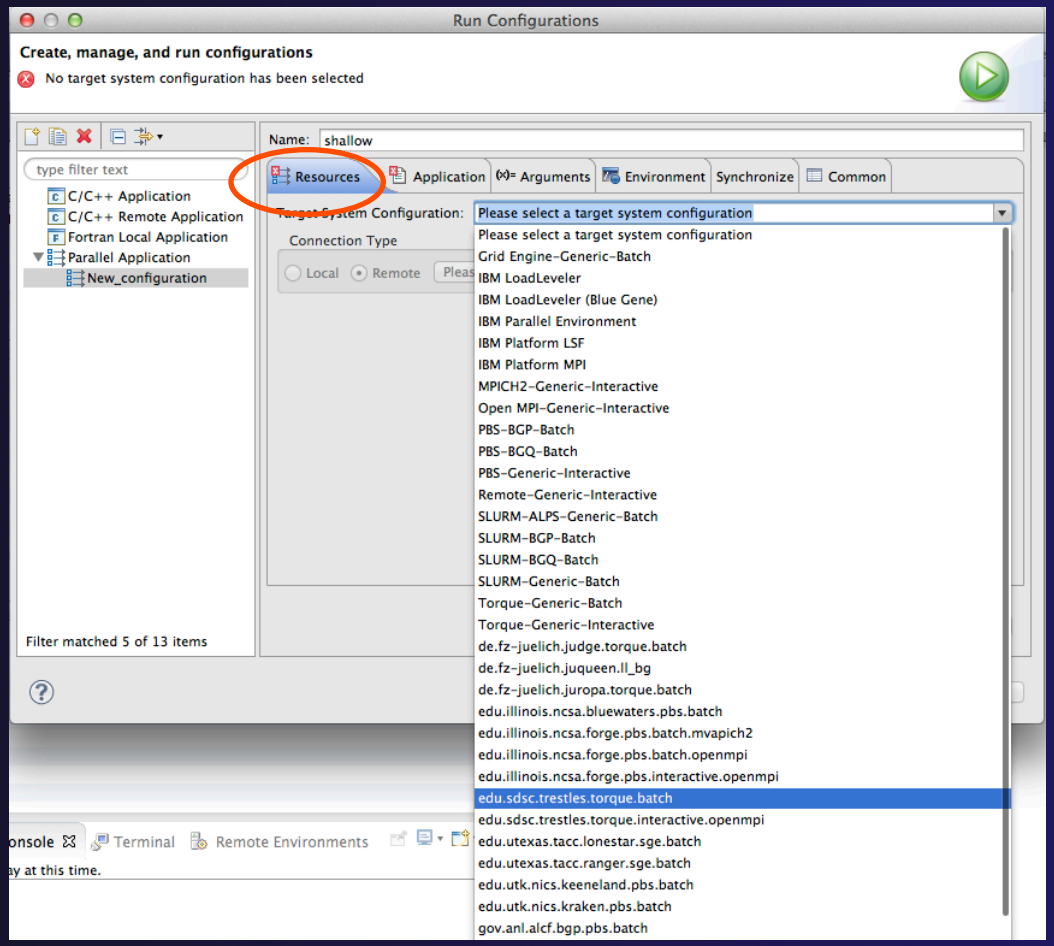
Set Run Configuration Name

- 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



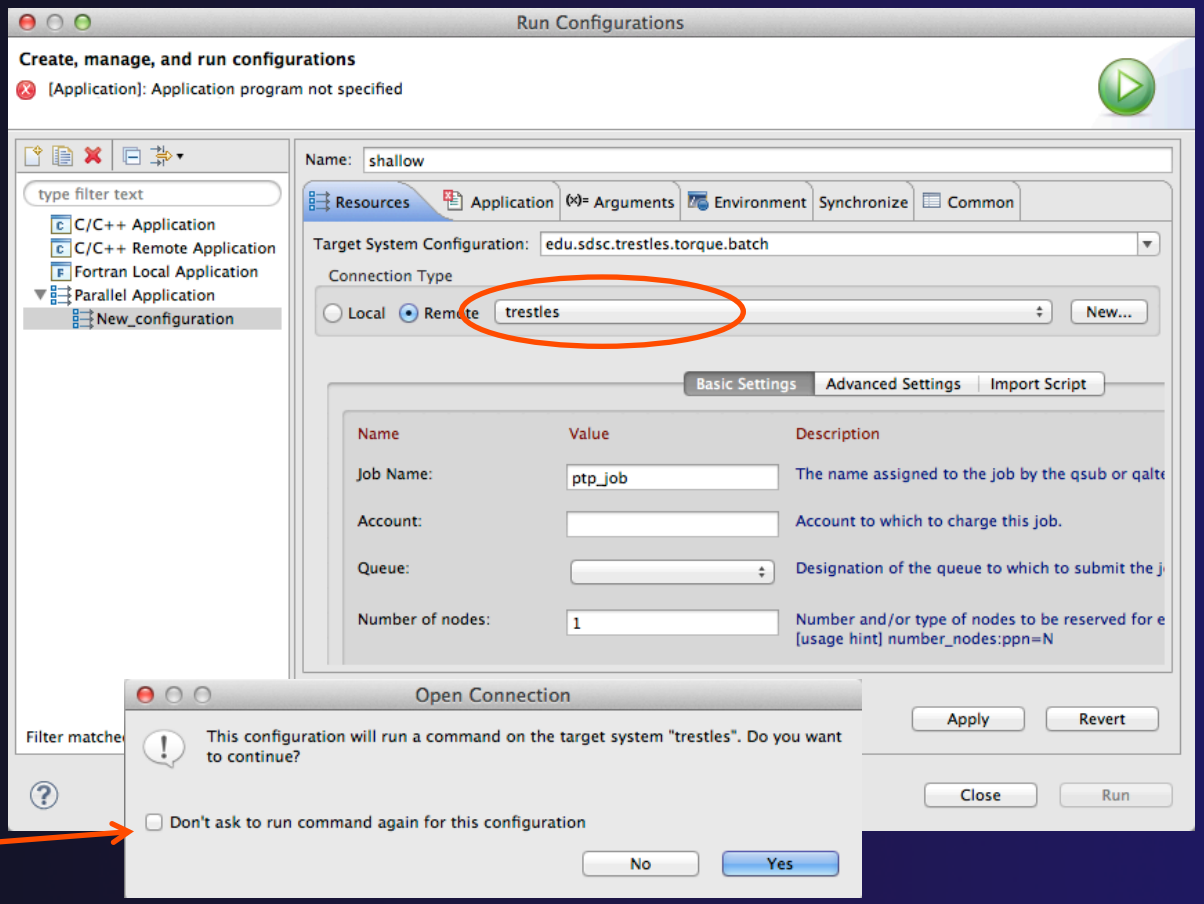
Configuring the Target System

- 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



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



Resources Tab

- 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*

Name: shallow

Resources | Application | Arguments | Environment | Synchronize | Common

Target System Configuration: edu.sdsc.trestles.torque.batch

Connection Type

Local | Remote | trestles | New...

Basic Settings | Advanced Settings | Import Script

Name	Value	Description
Job Name:	ptp_job	The name assigned to the job by the qsub or qalter command.
Account:		Account to which to charge this job.
Queue:		Designation of the queue to which to submit the job.
Number of nodes:	1	Number and/or type of nodes to be reserved for exclusive use by the job. [usage hint] number_nodes:ppn=N
Total Memory Needed:		Maximum amount of memory used by all concurrent processes in the job.
Wallclock Time:	00:30:00	Maximum amount of real time during which the job can be in the running state.
MPI Command:		Which mpi command to use.
MPI Number of Processes:	1	the '-np' value [usually equals Nodes*ppn]
Export Environment:	☑	All variables in the qsub command's environment are to be exported to the batch job.
Modules to Load:	Configure...	Modules that will be loaded inside the job script.

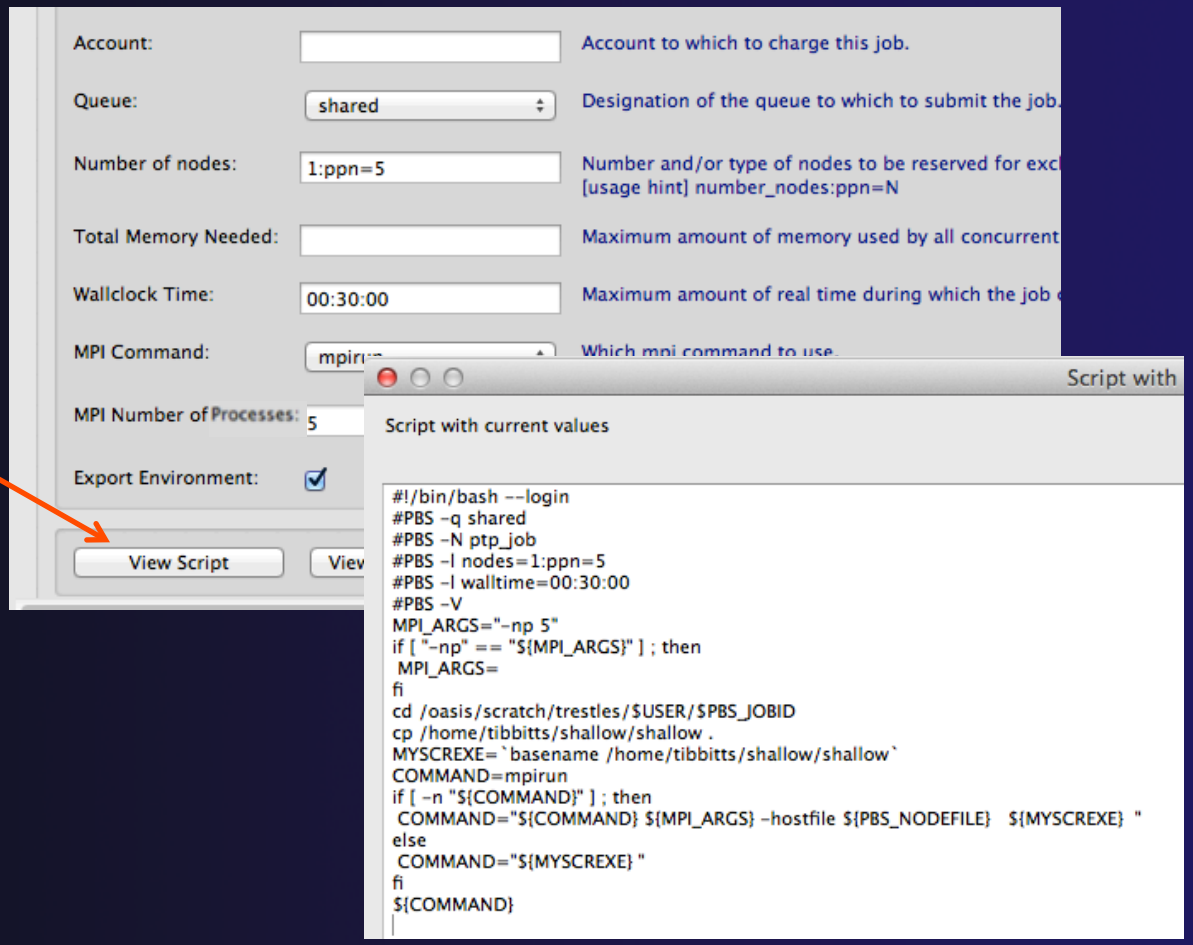
View Script | View Configuration | Restore Defaults

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

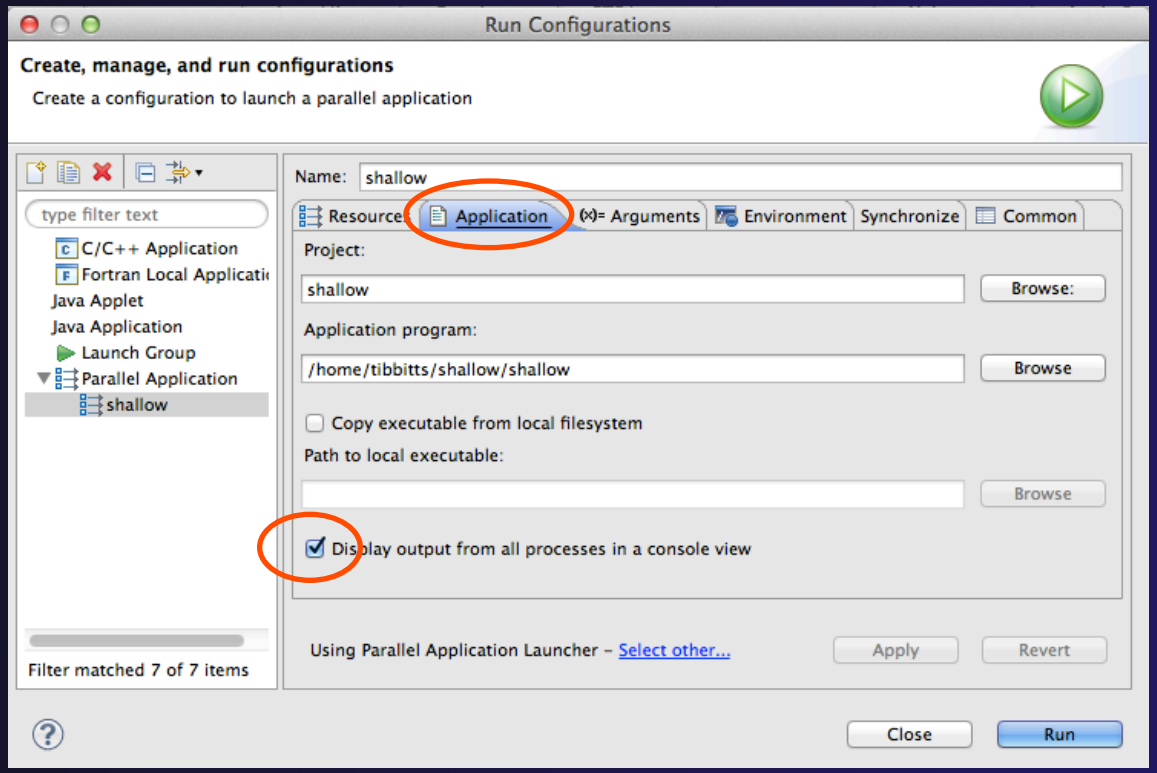
Viewing the Job Script

- 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



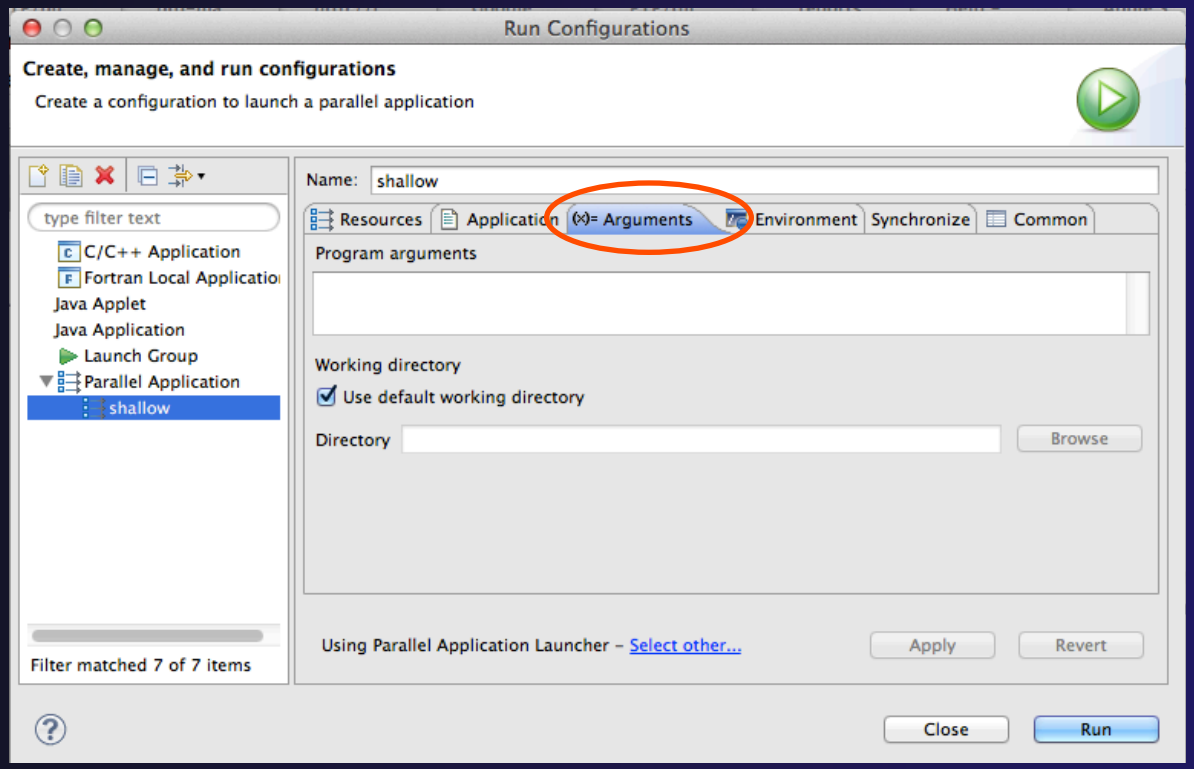
Application Tab

- 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**



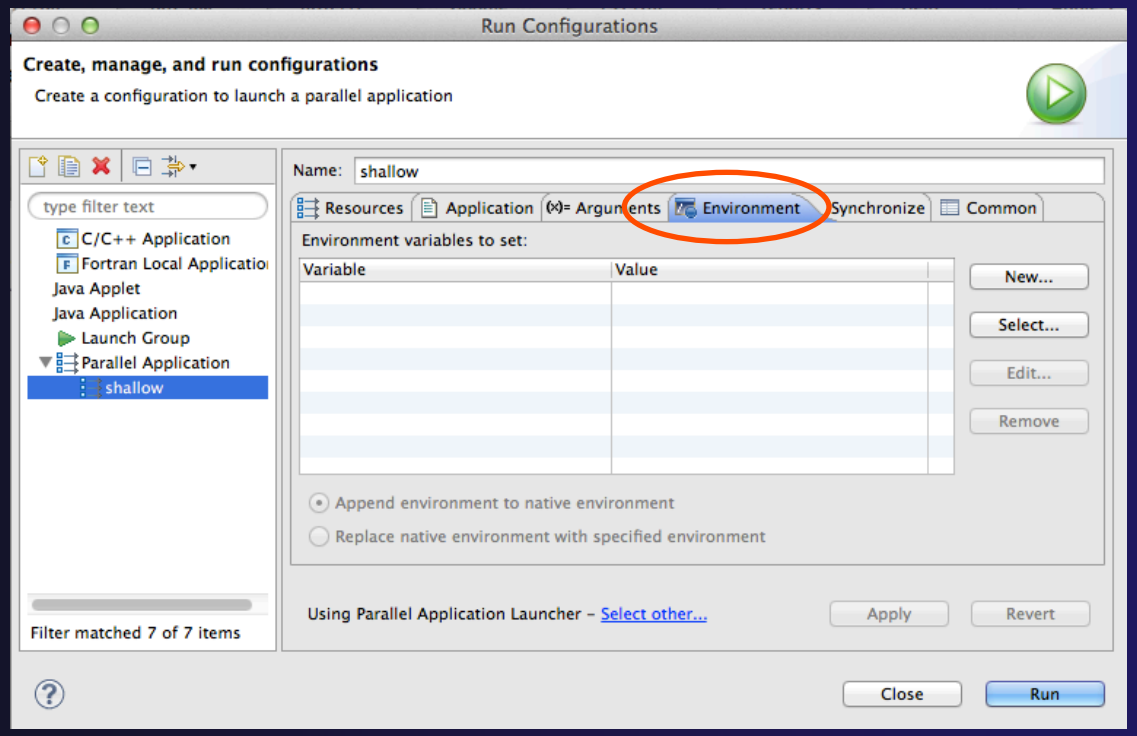
Arguments Tab (Optional)

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



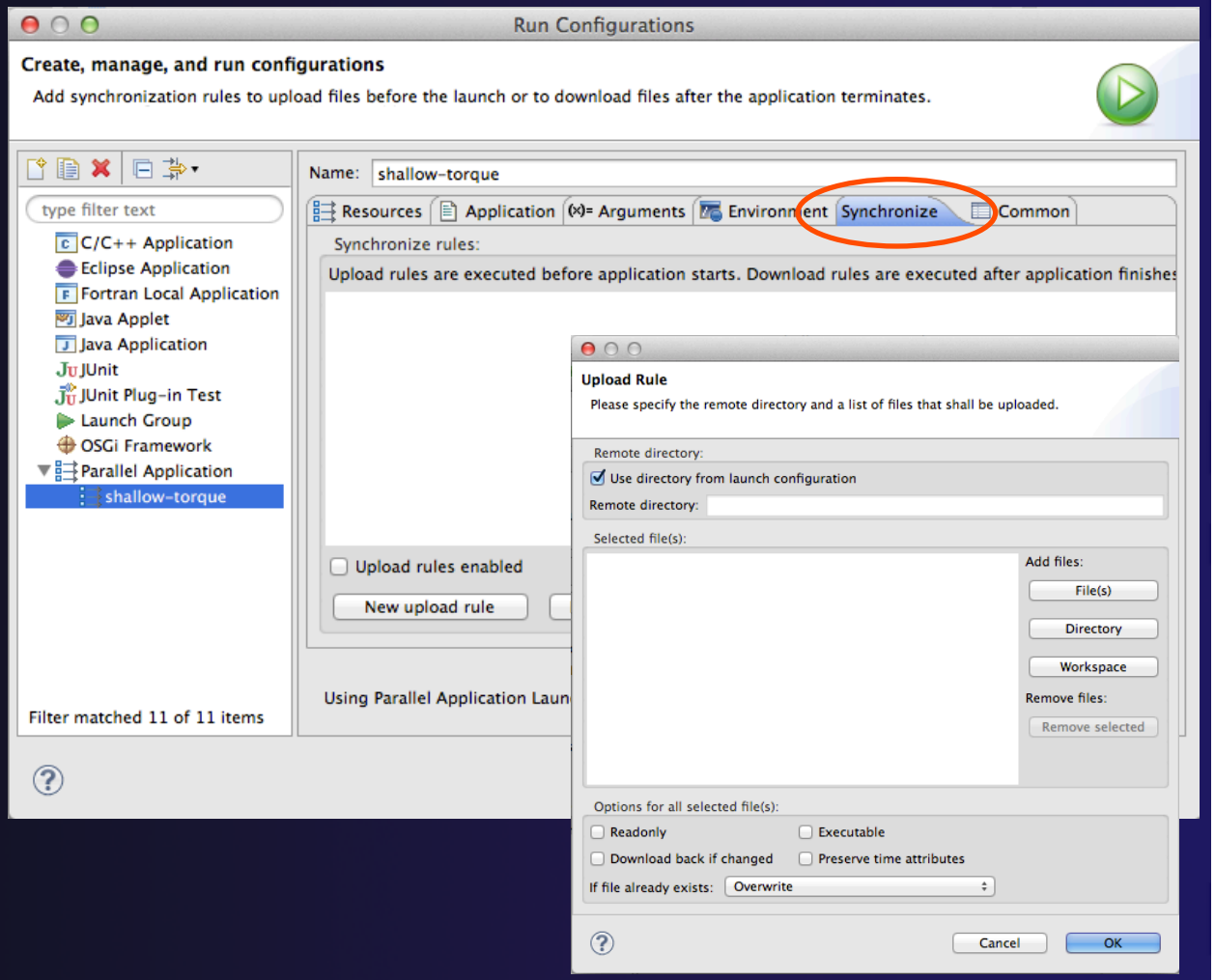
Environment Tab (Optional)

- 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



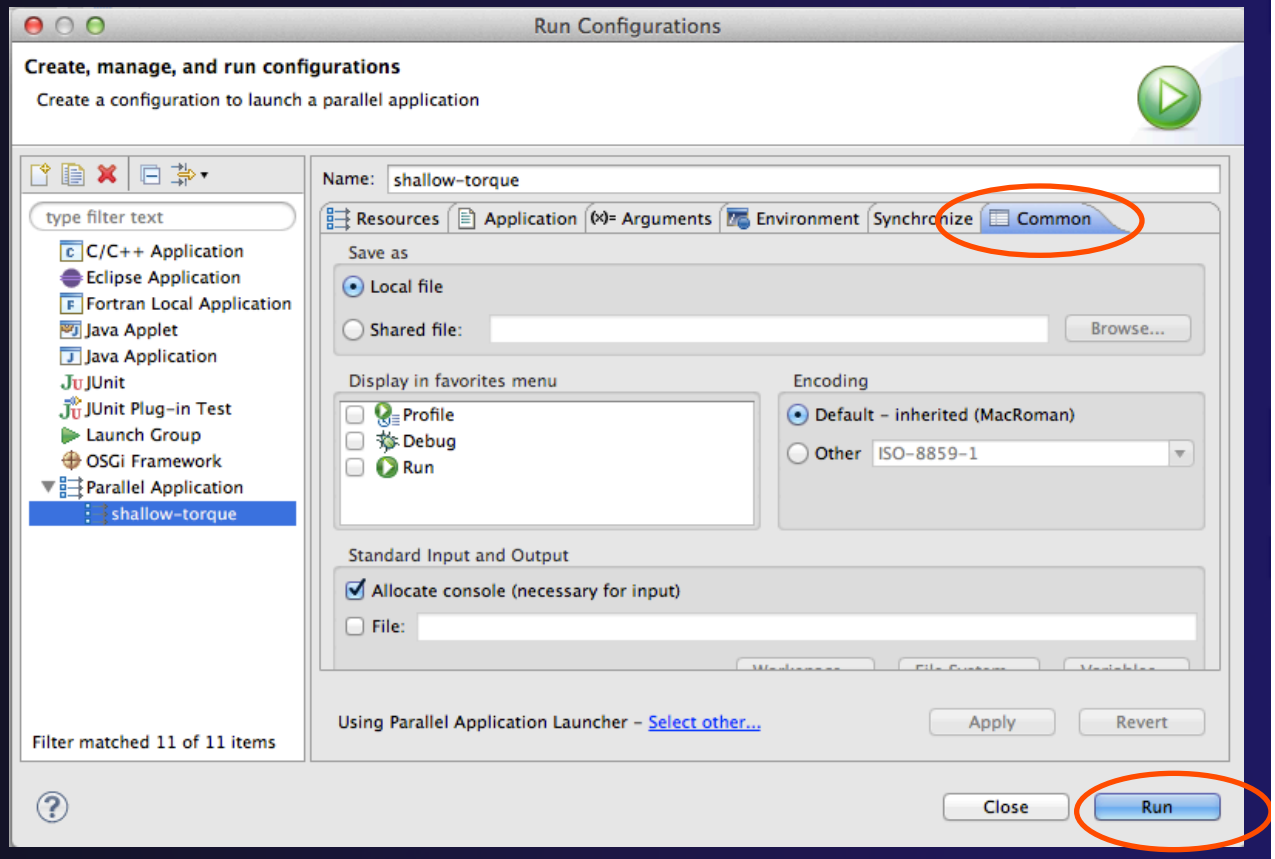
Synchronize Tab (Optional)

- 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



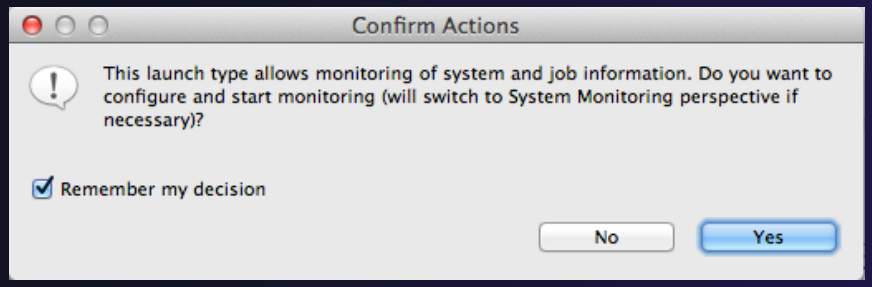
Common Tab (Optional)

- 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

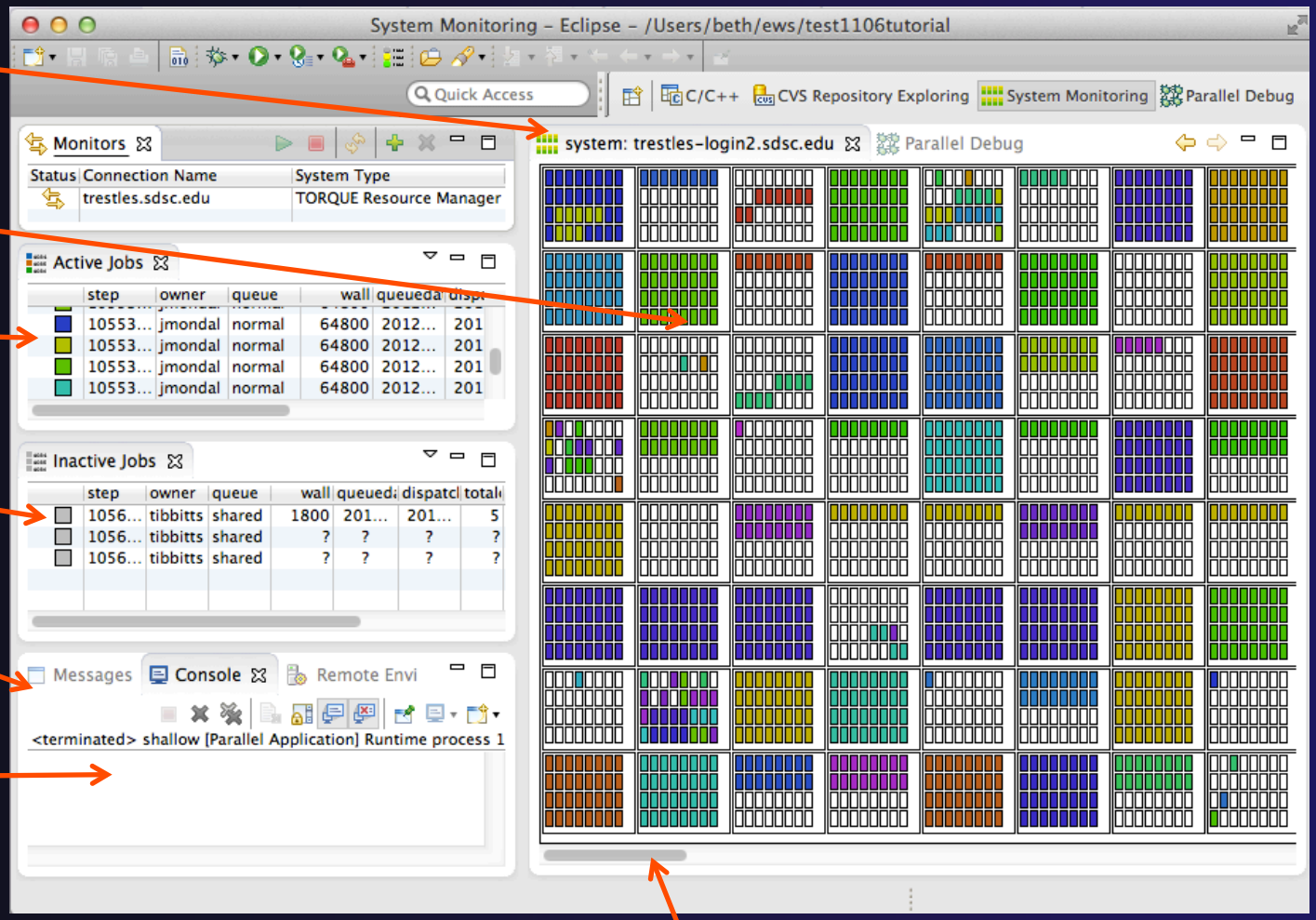
- Select **Run** to launch the job
- You may be asked to switch to the System Monitoring Perspective



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

System Monitoring Perspective

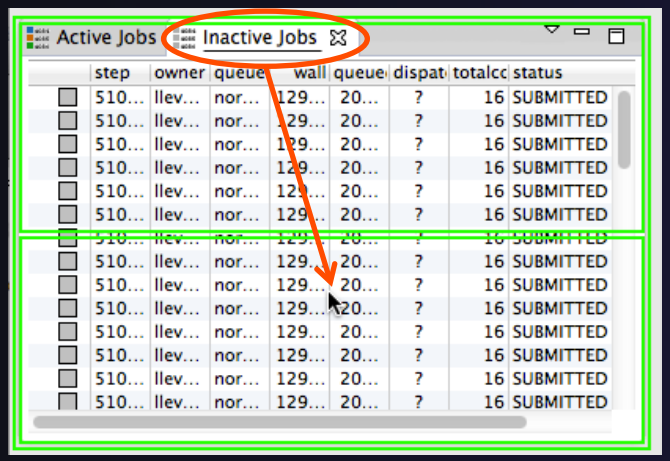
- System view
- Jobs running on system
- Active jobs
- Inactive jobs
- Messages
- Console

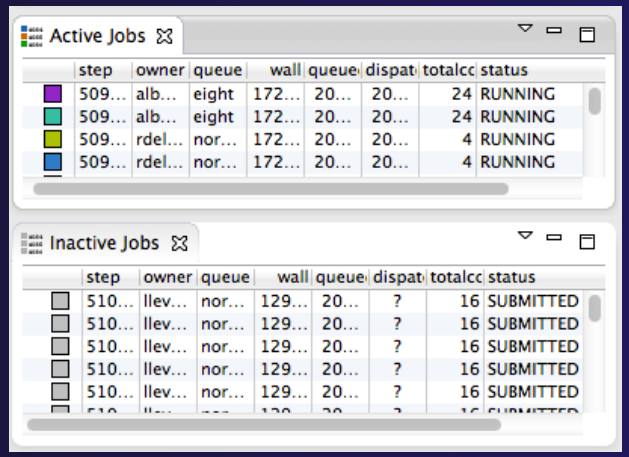


Scroll to see more

Moving views

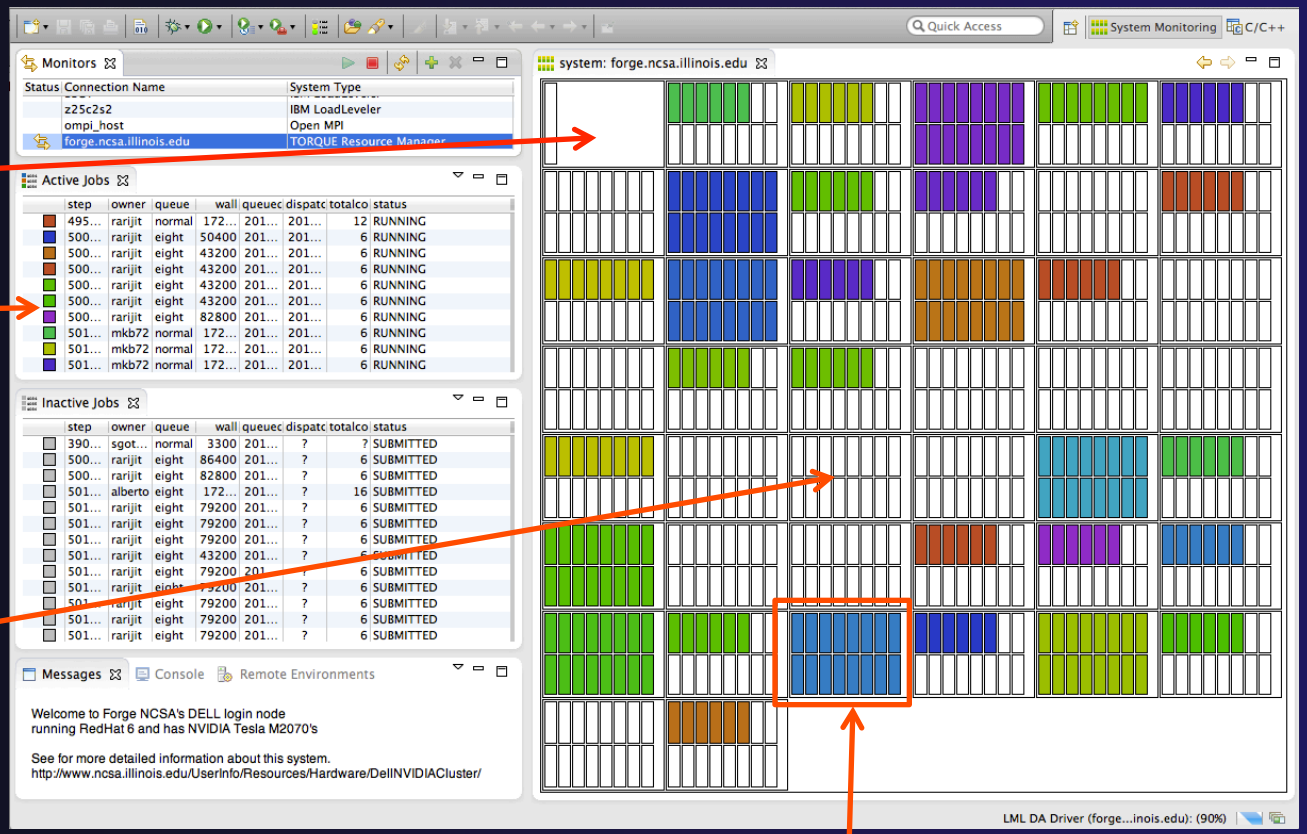
- The System Monitoring Perspective overlaps the **Active Jobs** and **Inactive Jobs** views
- To split them apart and see both at once, *drag* the tab for the **Inactive Jobs** view to the lower half of its area, and let go of mouse





System Monitoring

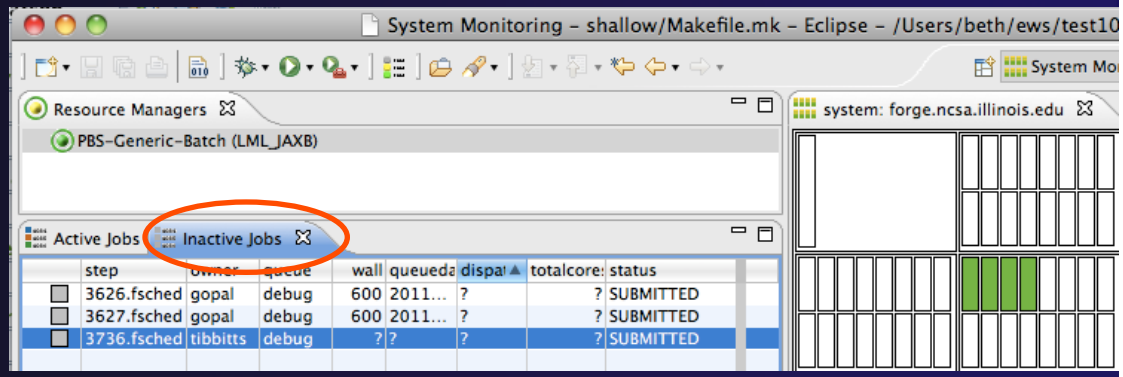
- **System** view, with abstraction of system configuration
- Hold mouse button down on a job in **Active Jobs** view to see where it is running in **System** view
- Hover over node in **System** view to see job running on node in **Active Jobs** view

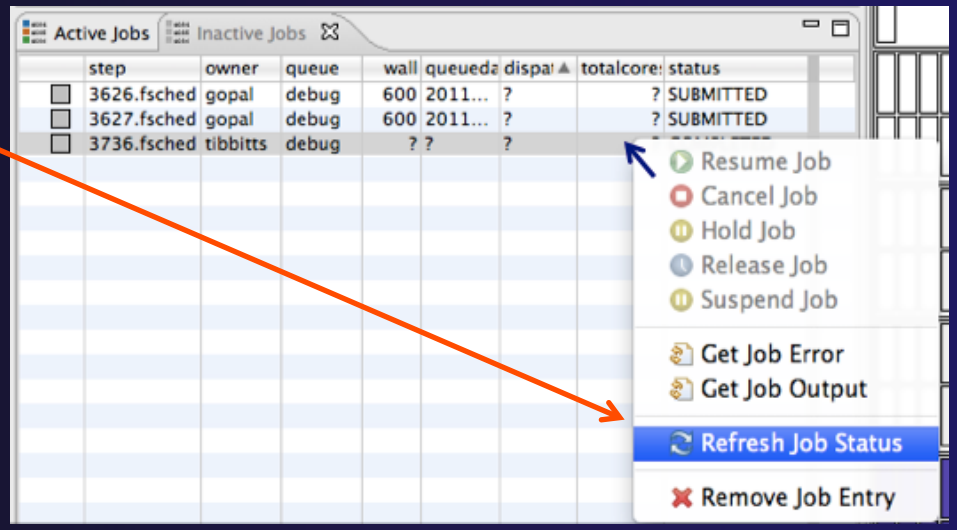


One node with 16 cores

Job Monitoring

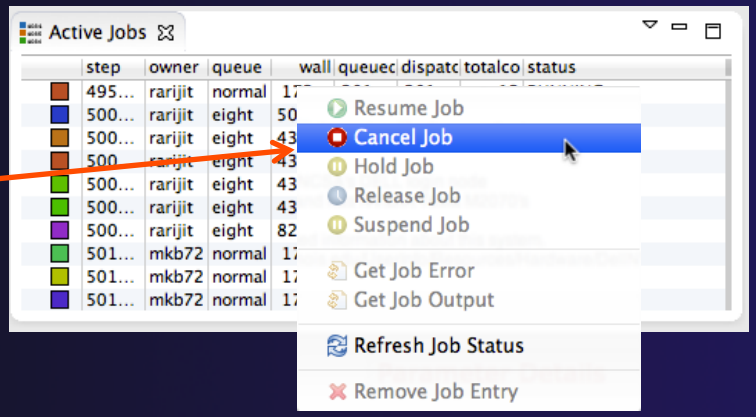
- Job initially appears in **Inactive Jobs** view
- Moves to the **Active Jobs** view when execution begings
- Returns to **Inactive Jobs** view on completion
- Status refreshes automatically every 60 sec
- Can force refresh with menu

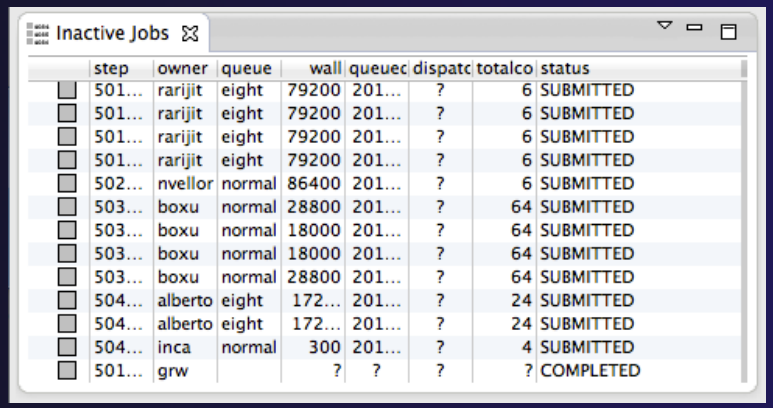




Controlling Jobs

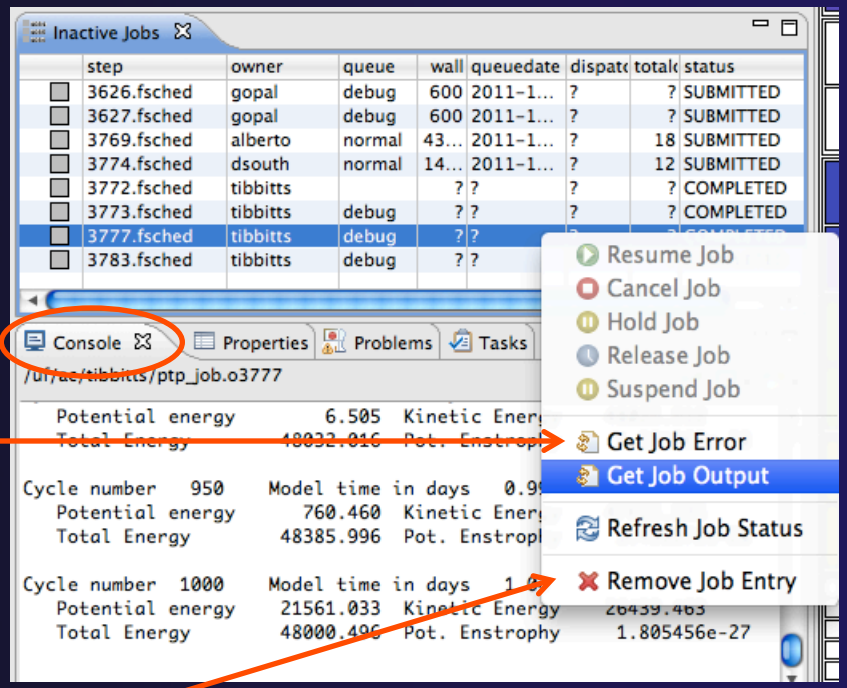
- Right click on a job to open context menu
- Actions will be enabled IFF
  - The job belongs to you
  - The action is available on the target system
  - The job is in the correct state for the action
- When job has COMPLETED, it will remain in the **Inactive Jobs** view





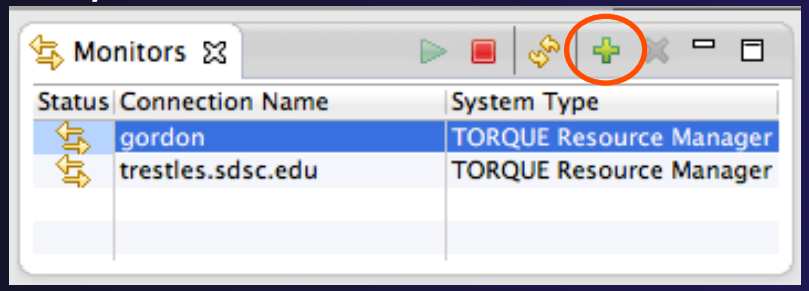
Obtaining Job Output

- 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**

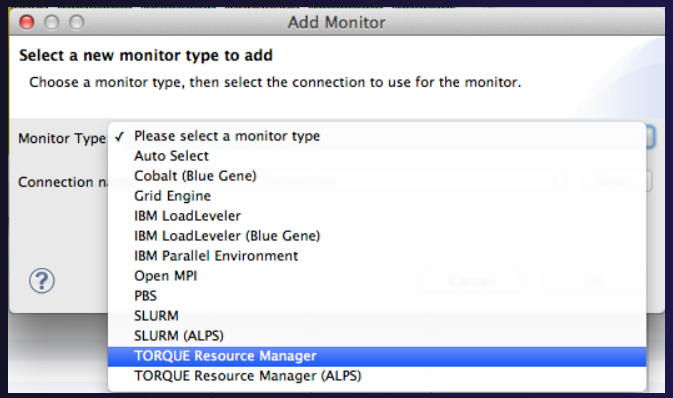


Add a Monitor

- 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

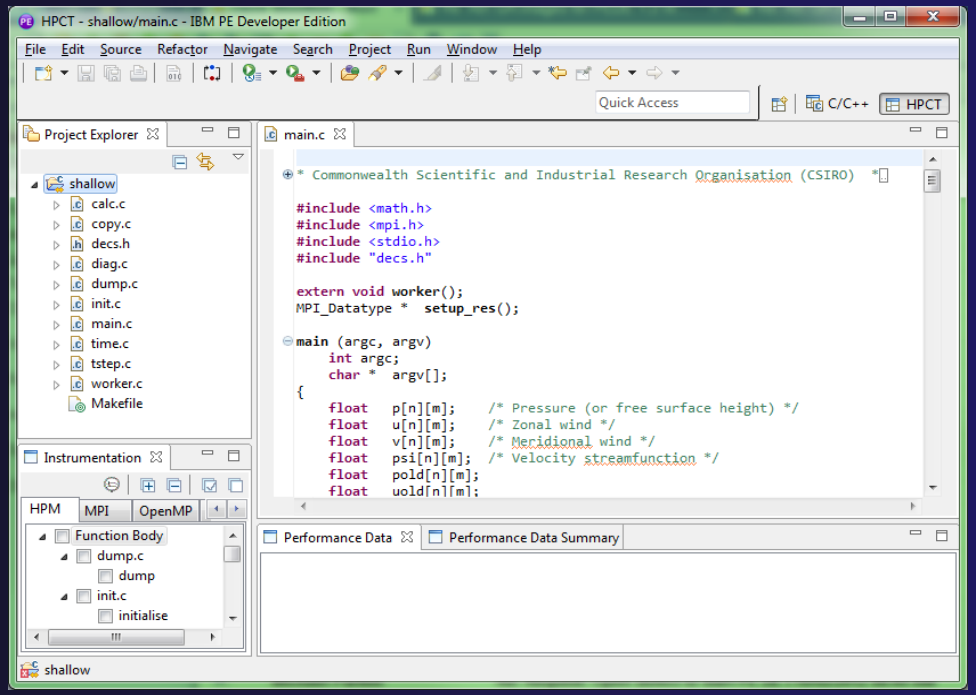
Exercise

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

Update May 2013

HPC Toolkit

- 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

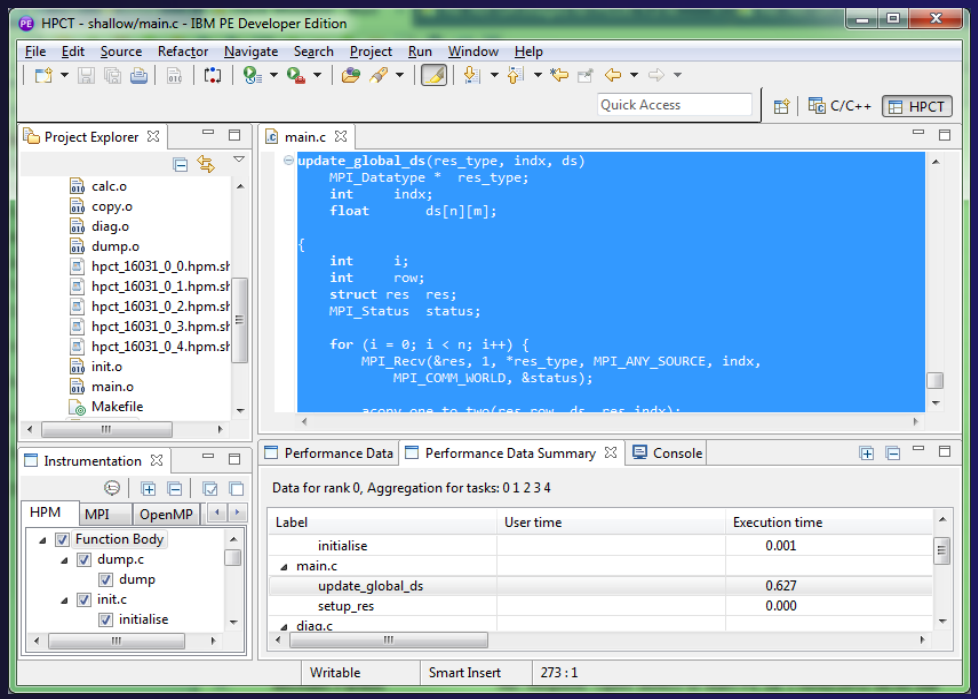


What is HPC Toolkit?

- 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

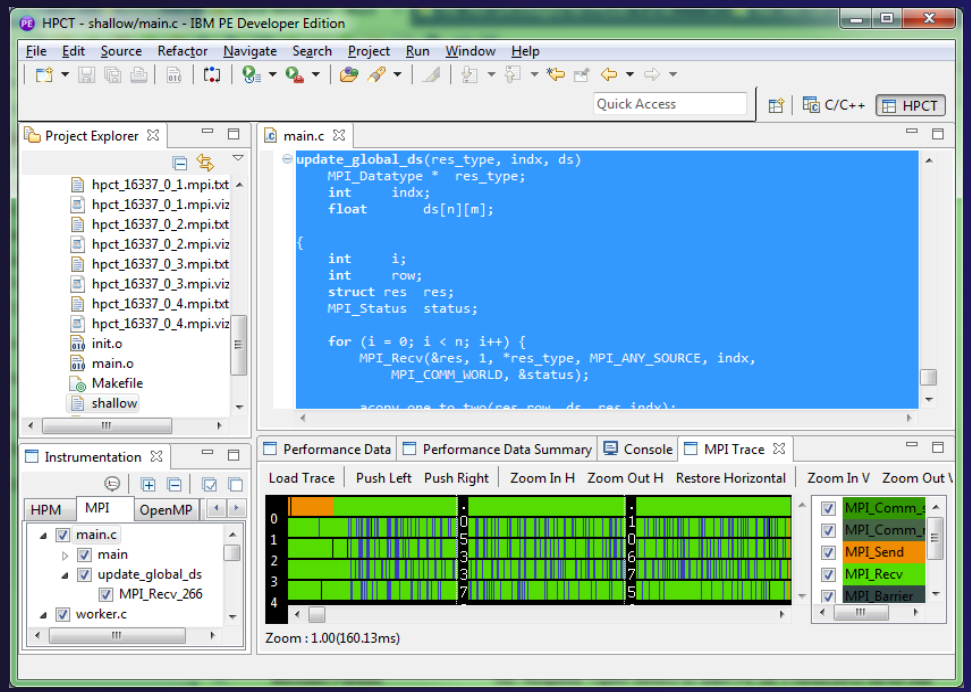
Hardware Performance Counters

- 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



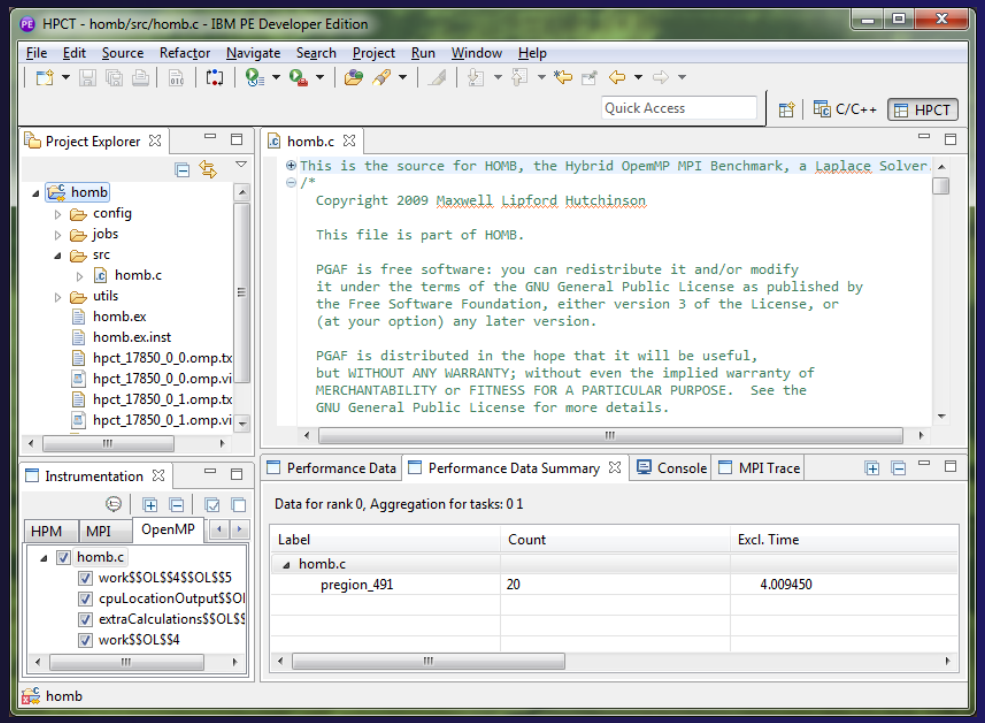
MPI Profiling and Tracing

- 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



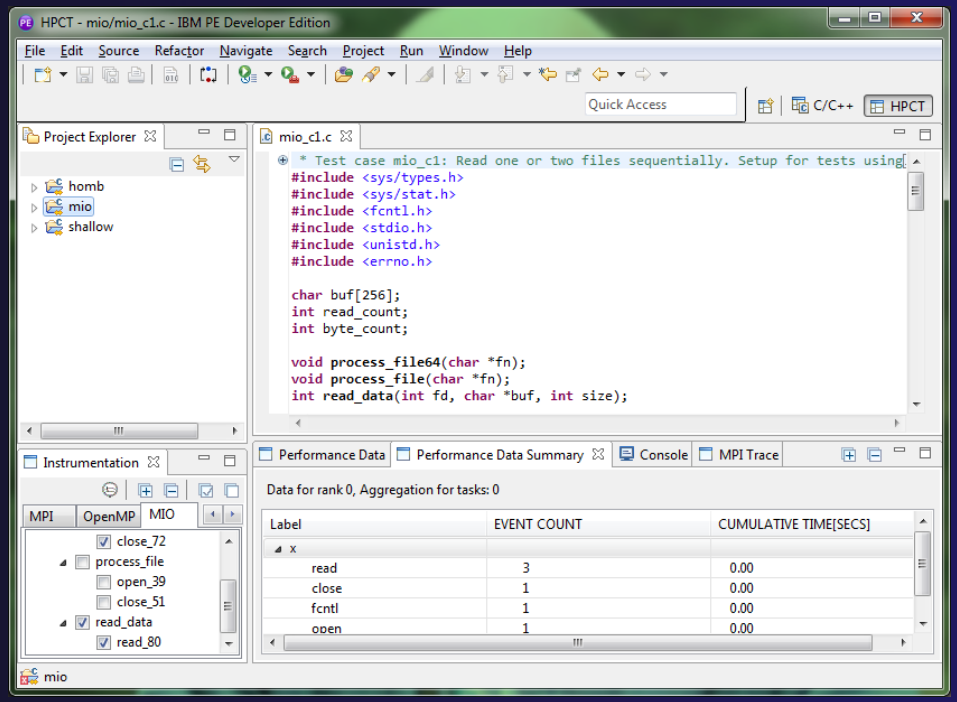
OpenMP Profiling

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

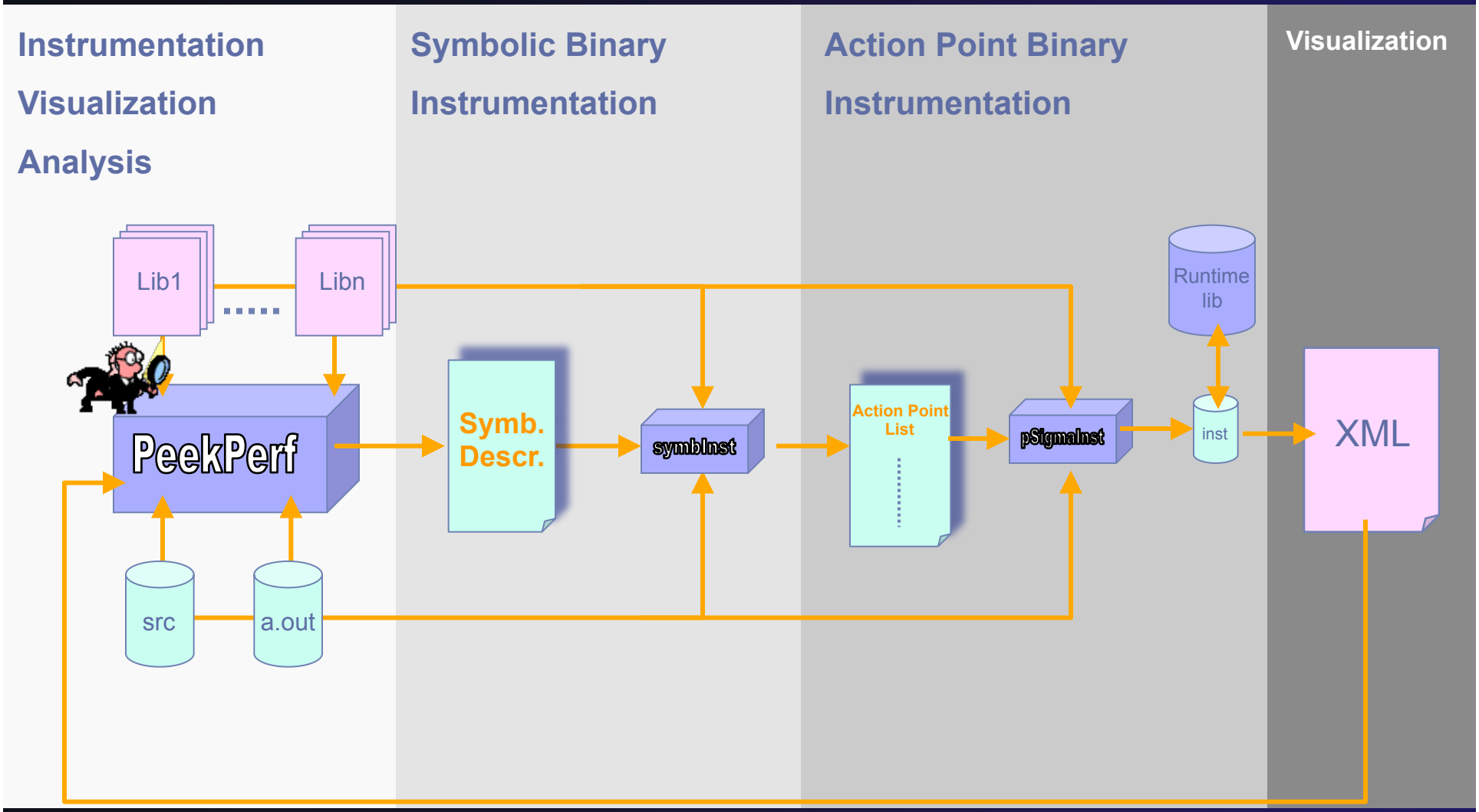


I/O Profiling and Tracing

- 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



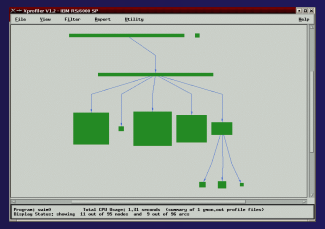
Binary Instrumentation



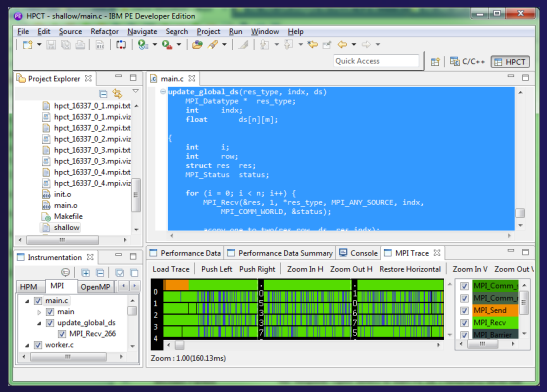
Using HPC Toolkit

Possible workflow

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



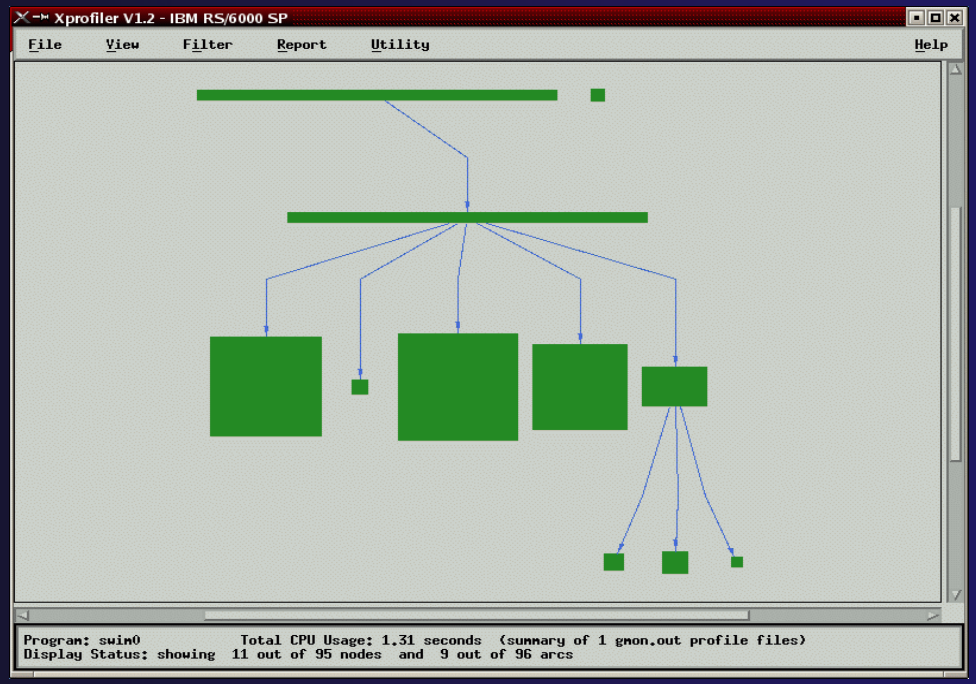
- Instrument 'hot' functions in application for performance data gathering
  - Hardware Performance Counters, MPI, OpenMP, I/O
- Run the instrumented application
- Analyze application performance with HPC Toolkit tools



- 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

HPCT - shallow/Makefile - IBM PE Developer Edition

File Edit Navigate Search Project Run Window Help

Quick Access | C/C++ | HPCT

Project Explorer

hpct_17587_0_0.mpi.viz
hpct_17587_0_1.mpi.txt
hpct_17587_0_1.mpi.viz
hpct_17587_0_2.mpi.txt
hpct_17587_0_2.mpi.viz
hpct_17587_0_3.mpi.txt
hpct_17587_0_3.mpi.viz
hpct_17587_0_4.mpi.txt
hpct_17587_0_4.mpi.viz
init.o
main.o
Makefile

Instrumentation

MPI | OpenMP | MIO

*Makefile

```
#CFLAGS =    -g -q64
CFLAGS =    -g -pg -q64
CC = mpcc

CFILES = calc.c copy.c diag.c init.c main.c time.c tstep.c worker.c dump.c
OBJECTS = calc.o copy.o diag.o init.o main.o time.o tstep.o worker.o dump.o
HEADERS = decs.h
#HFLAGS = -emit-stub-syms -Wl,--hash-style=sysv
HFLAGS =

all: shallow

clean:
    /bin/rm -f shallow $(OBJECTS) core
    /bin/rm -rf profdir.*

shallow: $(OBJECTS) $(HEADERS)
    uname
    $(CC) ${CFLAGS} ${HFLAGS} -o shallow $(OBJECTS) ${LIB} -lm

calc.o: calc.c decs.h
```

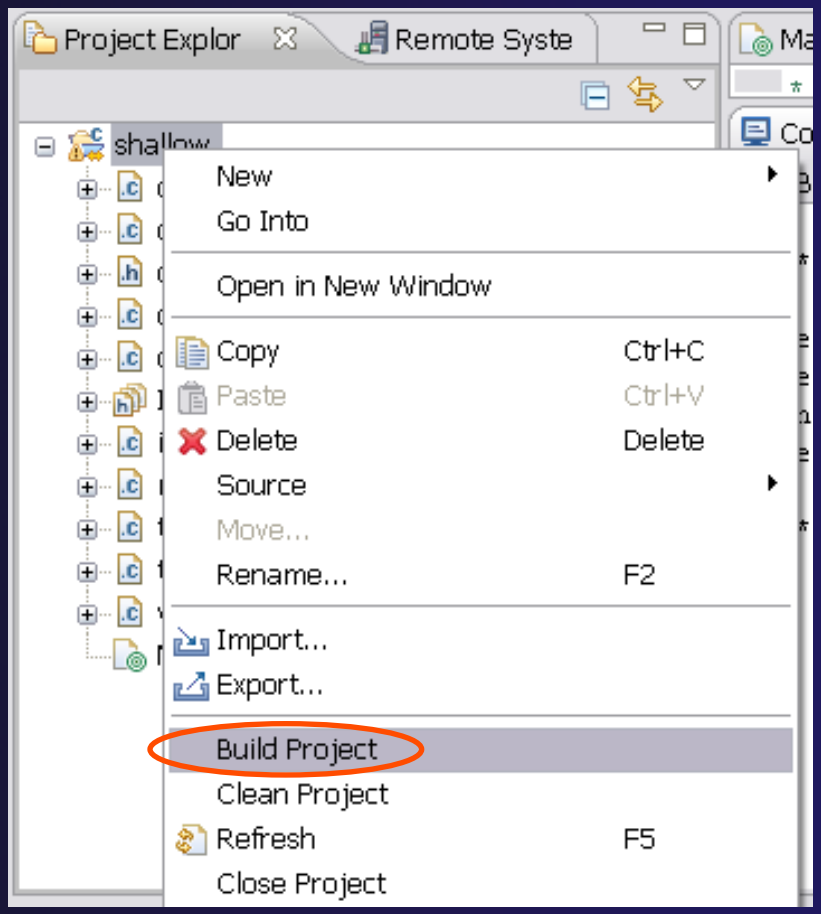
Performance Data | Performance Data Summary | Console | MPI Trace

Data for rank 0, Aggregation for tasks: 0

Label	EVENT COUNT	CUMULATIVE TIME[SECS]

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

Console

CDT Build Console [shallow]

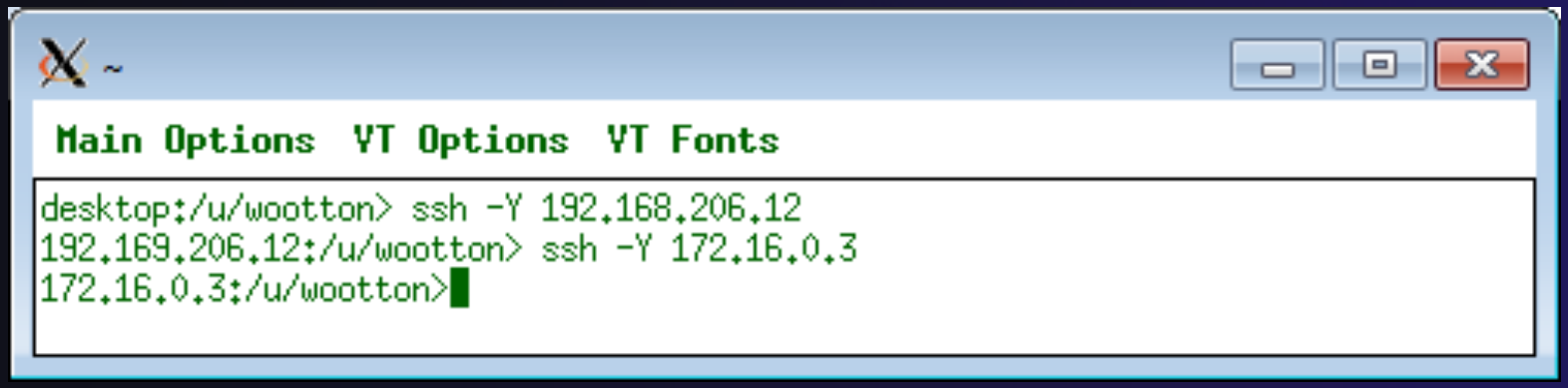
```
mpcc -g -pg   -c -o calc.o calc.c
mpcc -g -pg   -c -o copy.o copy.c
mpcc -g -pg   -c -o diag.o diag.c
mpcc -g -pg   -c -o init.o init.c
mpcc -g -pg   -c -o main.o main.c
mpcc -g -pg   -c -o time.o time.c
mpcc -g -pg   -c -o tstep.o tstep.c
mpcc -g -pg   -c -o worker.o worker.c
mpcc -g -pg   -c -o dump.o dump.c
mpcc -g -pg  -o shallow calc.o copy.o diag.o init.o main.o time.o tstep.o worker.o dump.o -lm
```

Project Explorer

- shallow
  - calc.c
  - copy.c
  - decs.h
  - diag.c
  - dump.c
  - Includes
  - init.c
  - main.c
  - time.c
  - tstep.c
  - worker.c
  - calc.o
  - copy.o
  - diag.o
  - dump.o
  - init.o
  - main.o
  - Makefile
  - shallow
  - time.o
  - tstep.o
  - worker.o

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)



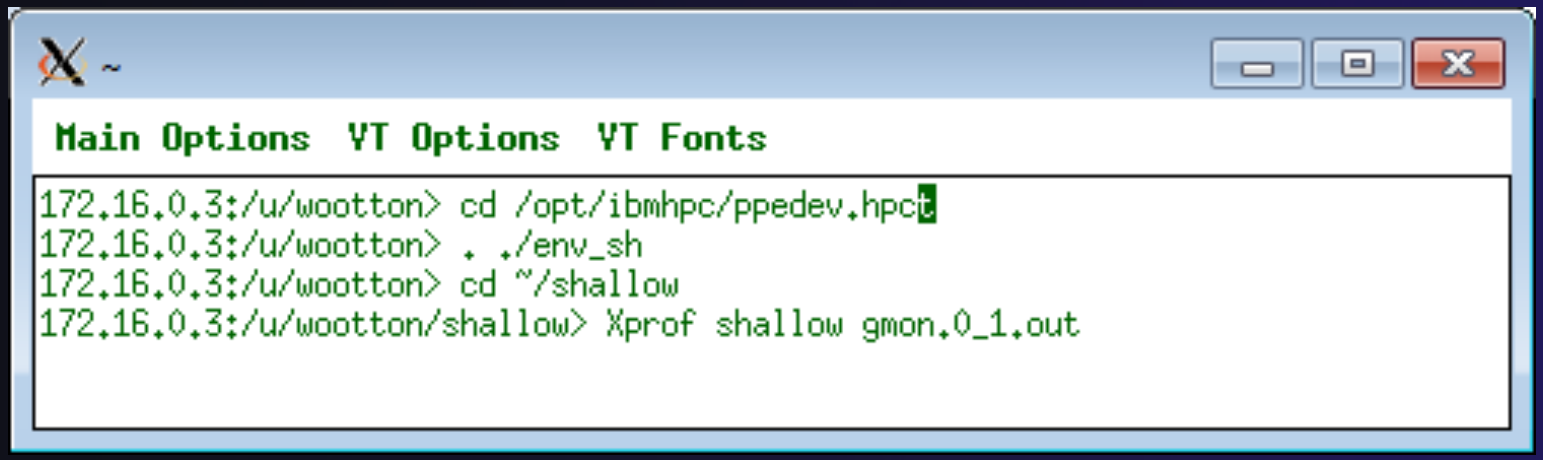
Set Up and Run Application

- Set PE environment variables
- Run application

```
~
Main Options  VT Options  VT Fonts
export MP_PROCS=5
export MP_TASKS_PER_NODE=5
export MP_EUILIB=us
export MP_RESD=yes
export MP_RMPOOL=1
export MP_EUIDEVICE=sn_single
export MP_DEVTYPE=hfi
$ cd ~/shallow
$ ./shallow
```

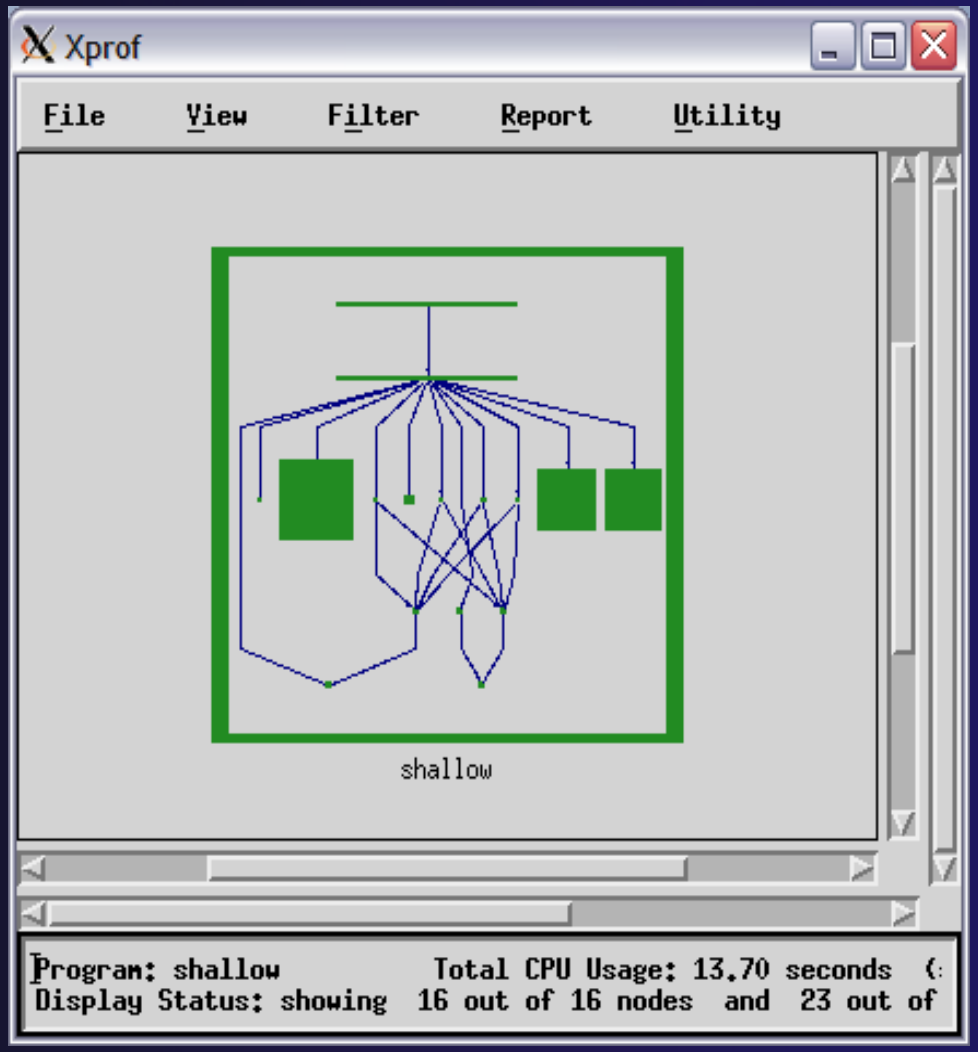
Display results using Xprof

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



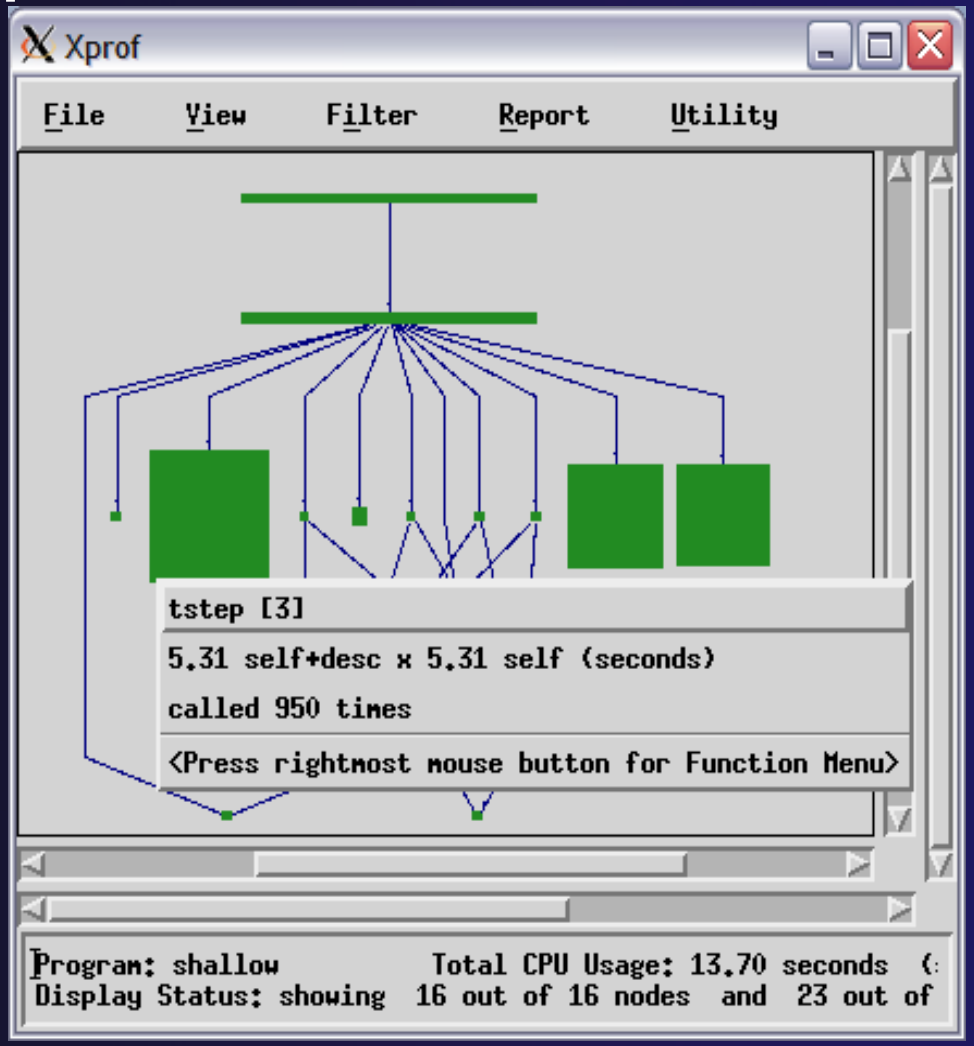
Xprof Initial View

- 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



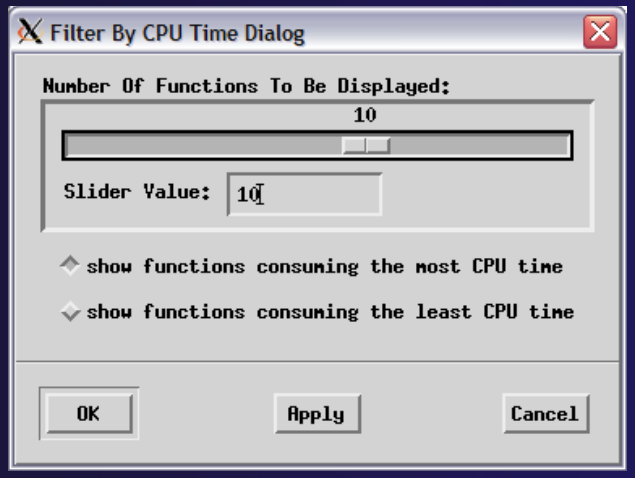
Expand Application Tree

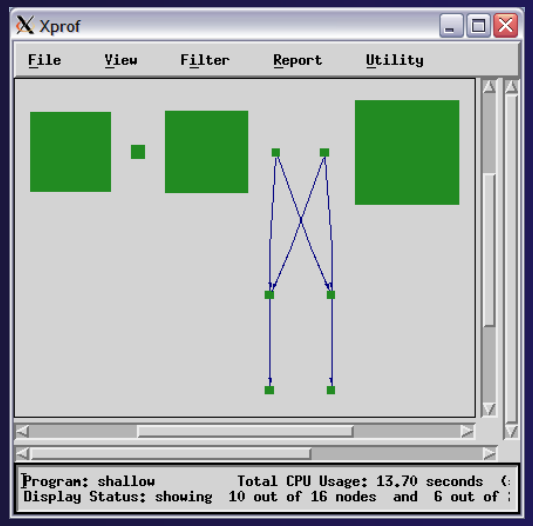
- Right click on outer green box
- Click **Remove Cluster Box** in popup menu
- Left click on nodes (boxes) to see function detail
- Left click on arcs to see caller/callee details
- Right click node or arc for additional actions
- Use **View** menu to zoom
- Use scrollbars to scroll



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

Call Graph Profile

File

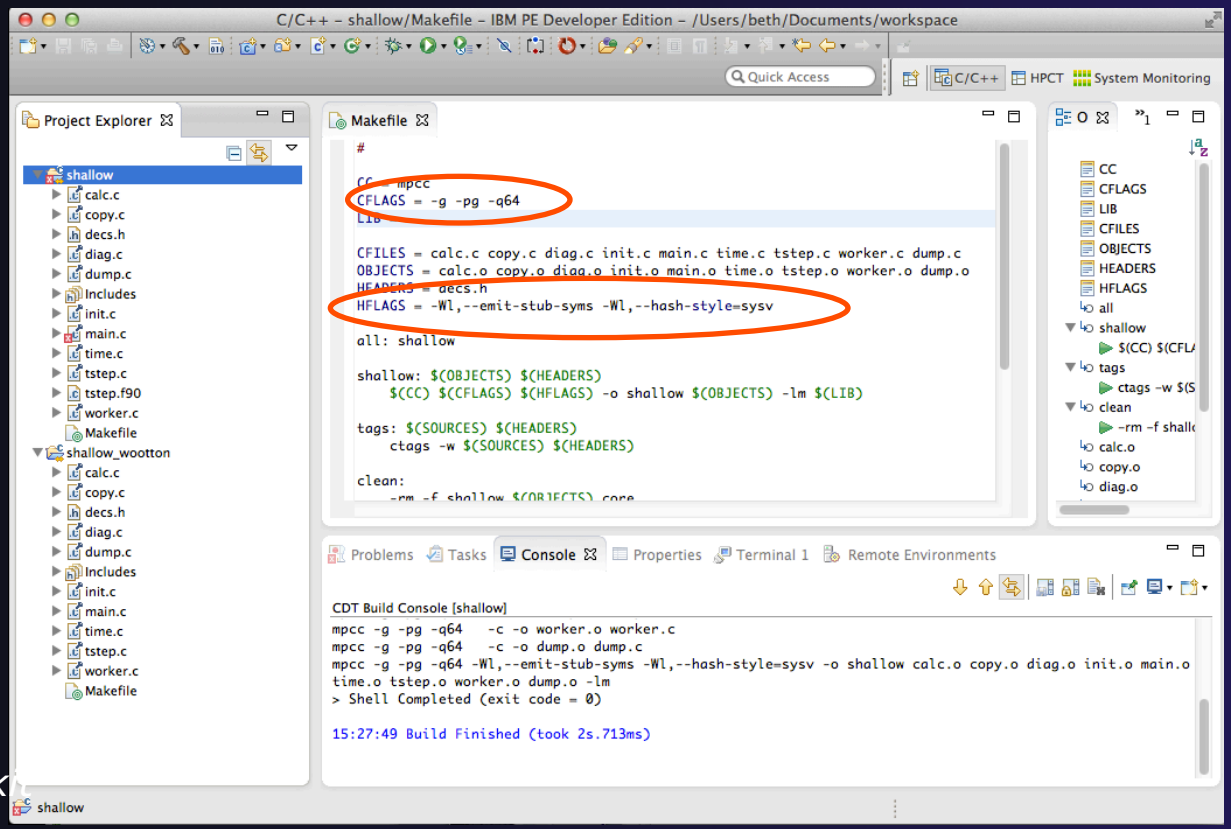
```
                                  called/total       parents
index  %time    self descendents  called+self    name           index
                                  called/total       children

                0.00       13.70       1/1           main [2]
[1]    100.0    0.00       13.70       1         worker [1]
                5.31        0.00     950/950         tstep [3]
                4.10        0.00     950/950         timetend [4]
                4.00        0.00     950/950         calcuvzh [5]
                0.29        0.00      19/19          diag [6]
                0.00        0.00    3584/13084       acopy_one_to_two [7]
                0.00        0.00     950/950         calc_load [11]
                0.00        0.00     950/950         calc_unload [12]
                0.00        0.00     950/950         time_load [13]
                0.00        0.00     950/950         time_unload [14]
                0.00        0.00       5/5           send_updated_ds [15]
                0.00        0.00       1/1           setup_res [16]

-----------------------------------------------
                                                   <spontaneous>
[2]    100.0    0.00       13.70                 main [2]
                0.00       13.70       1/1           worker [1]

-----------------------------------------------
                5.31        0.00     950/950         worker [1]
[3]     38.8    5.31        0.00     950         tstep [3]

-----------------------------------------------
                4.10        0.00     950/950         worker [1]
[4]     29.9    4.10        0.00     950         timetend [4]

-----------------------------------------------
                4.00        0.00     950/950         worker [1]
[5]     29.2    4.00        0.00     950         calcuvzh [5]
```

Search Engine: (regular expressions supported)

Hardware Performance Counter Profiling

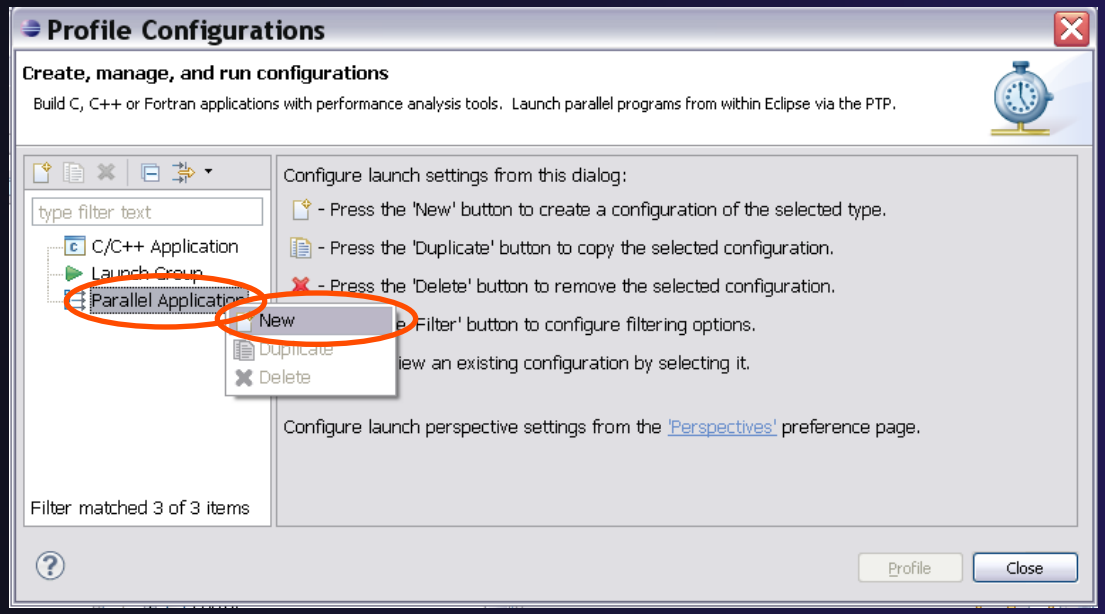
Rebuild Application

- 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**



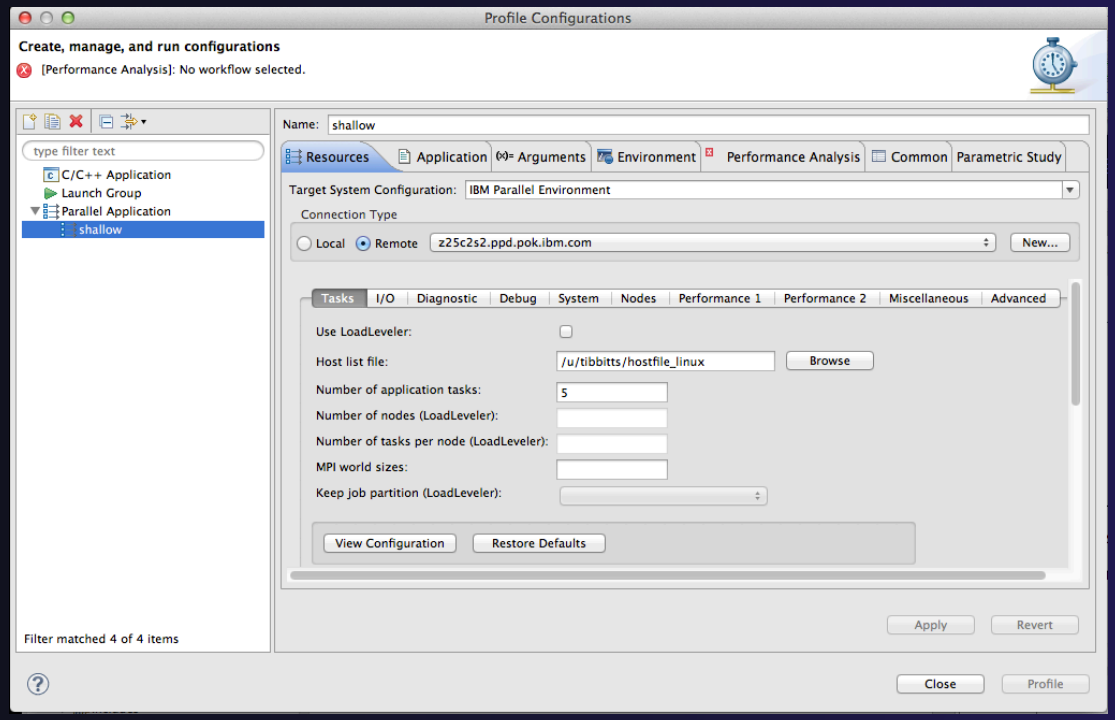
Set up Profile Configuration

- 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

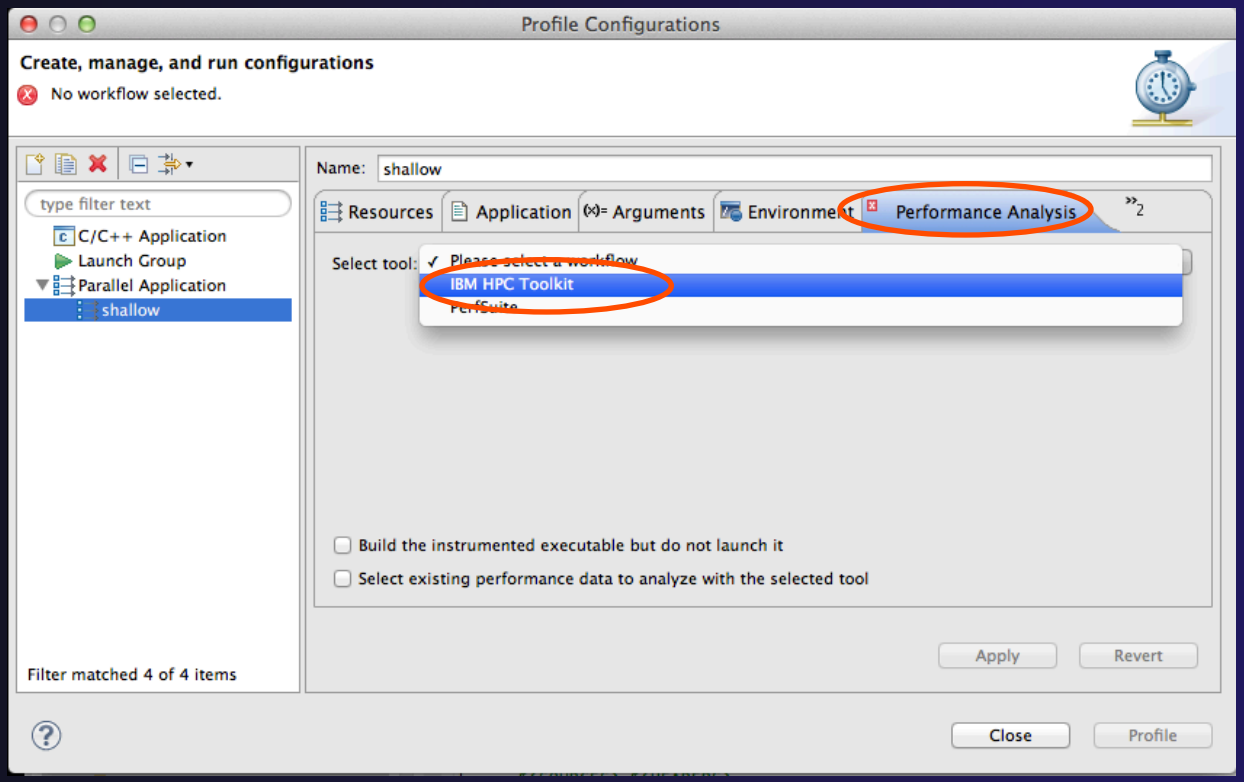
- Launch config information from previous run



Set up Profile Configuration 3

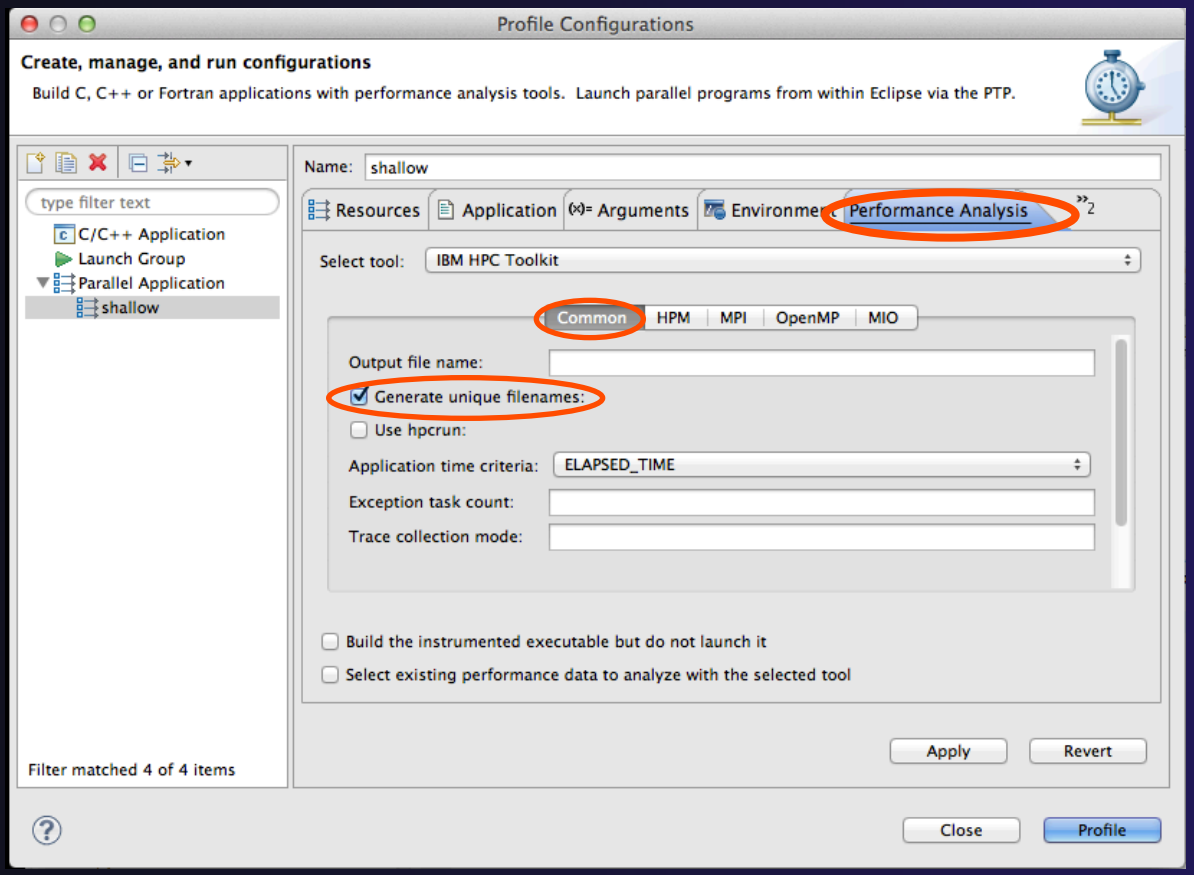
- Switch to **Performance Analysis** tab
- If not already set, **Select Tool: HPC Toolkit**

Wait while the executable is analyzed "Analyzing Binary"



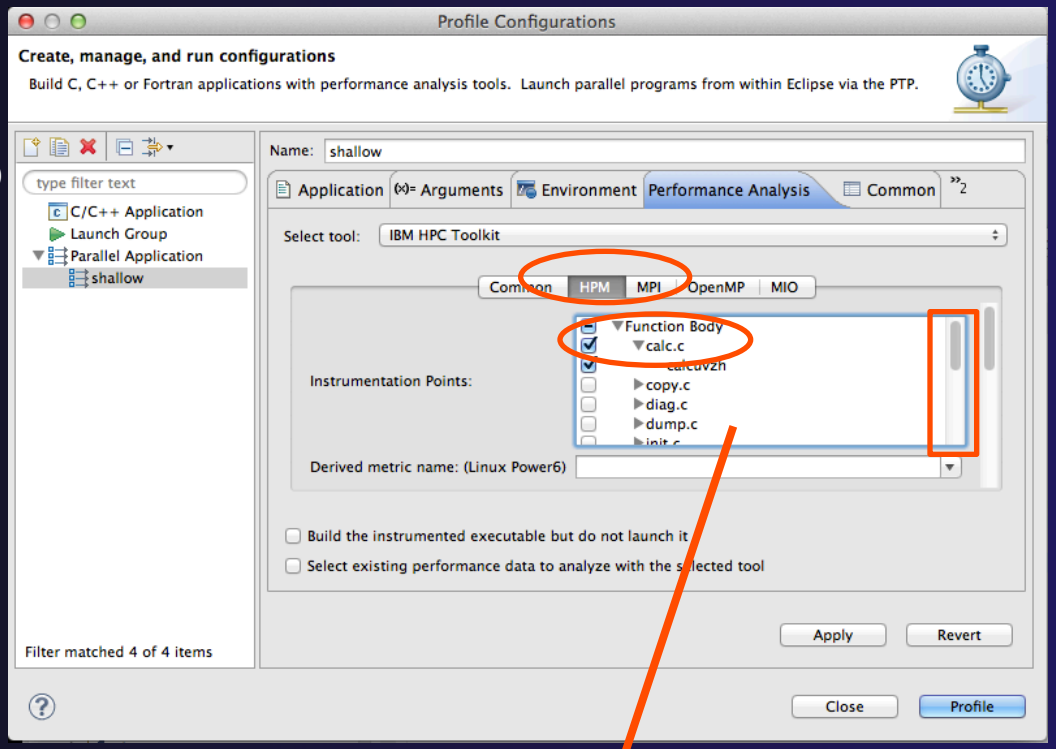
Set up Profile Configuration 4

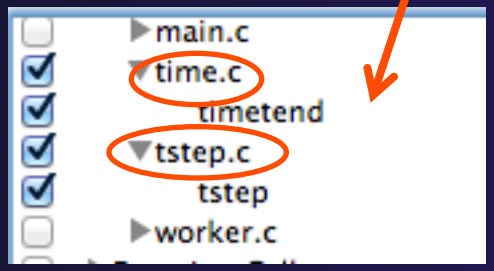
- Select the **Common** tab
- Check the **Generate unique filenames** checkbox
  - Adds job PID to performance output filenames



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

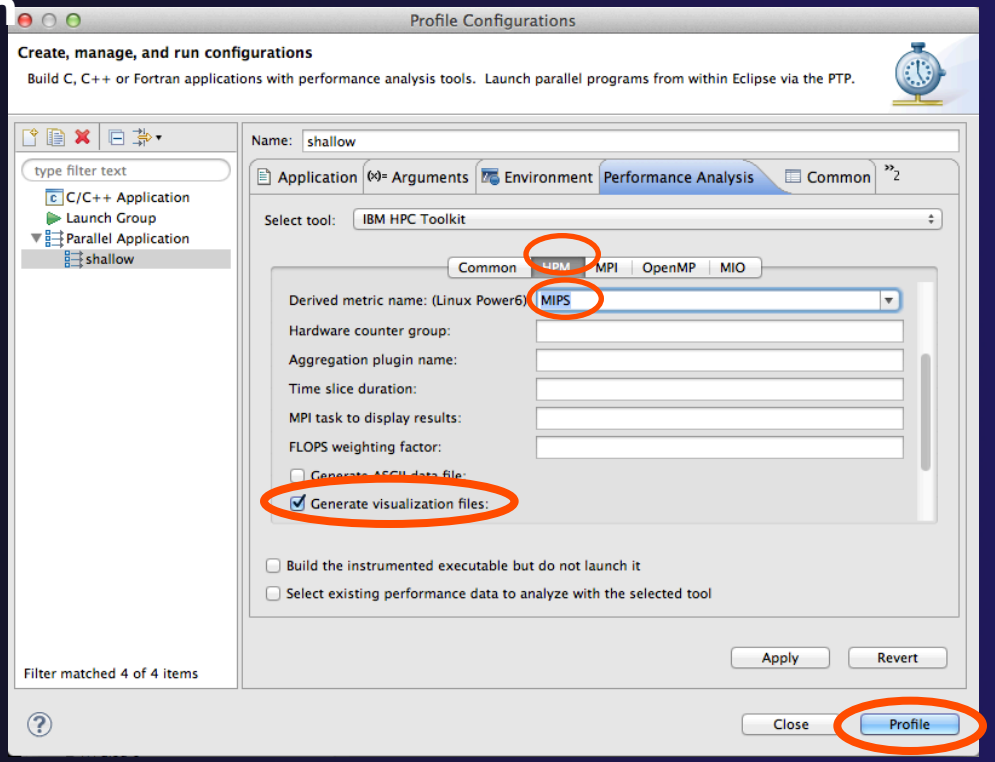




Scroll to select more

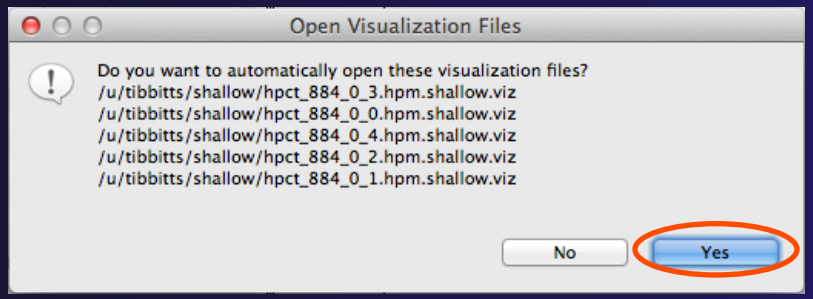
Profile Configuration: HPM 2

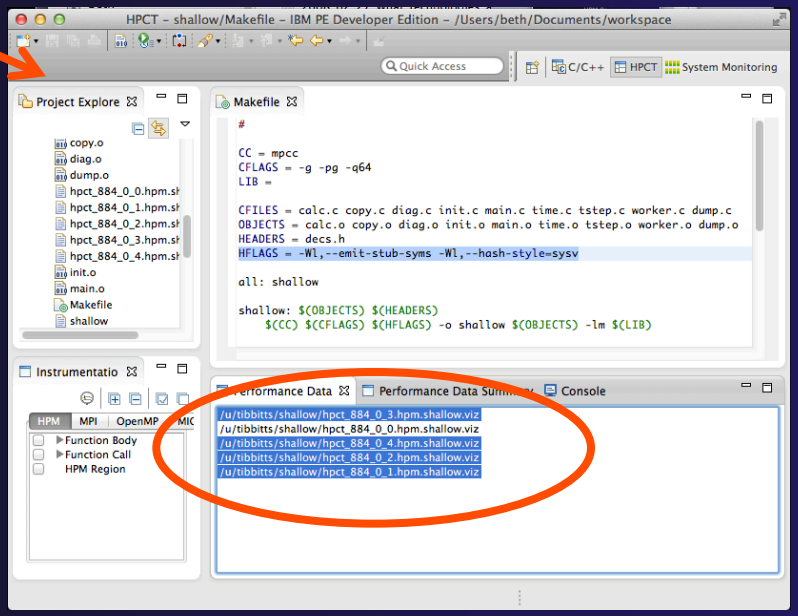
- Select **Derived Metric Name** (e.g. **MIPS)**
- Select **Generate visualization files**
- Select **Profile button** to run the profile analysis
- Program is instrumented then executed



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

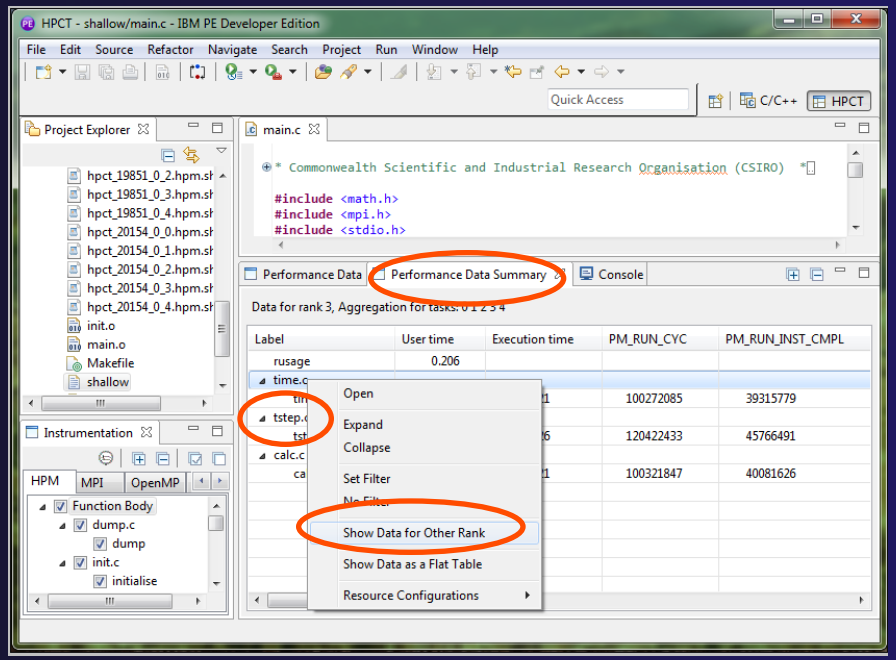


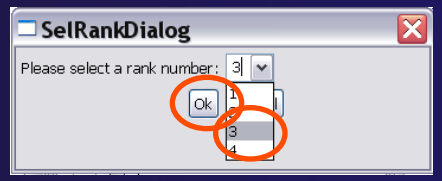


View HPM Performance Data 2

Open the **Performance Data Summary** view

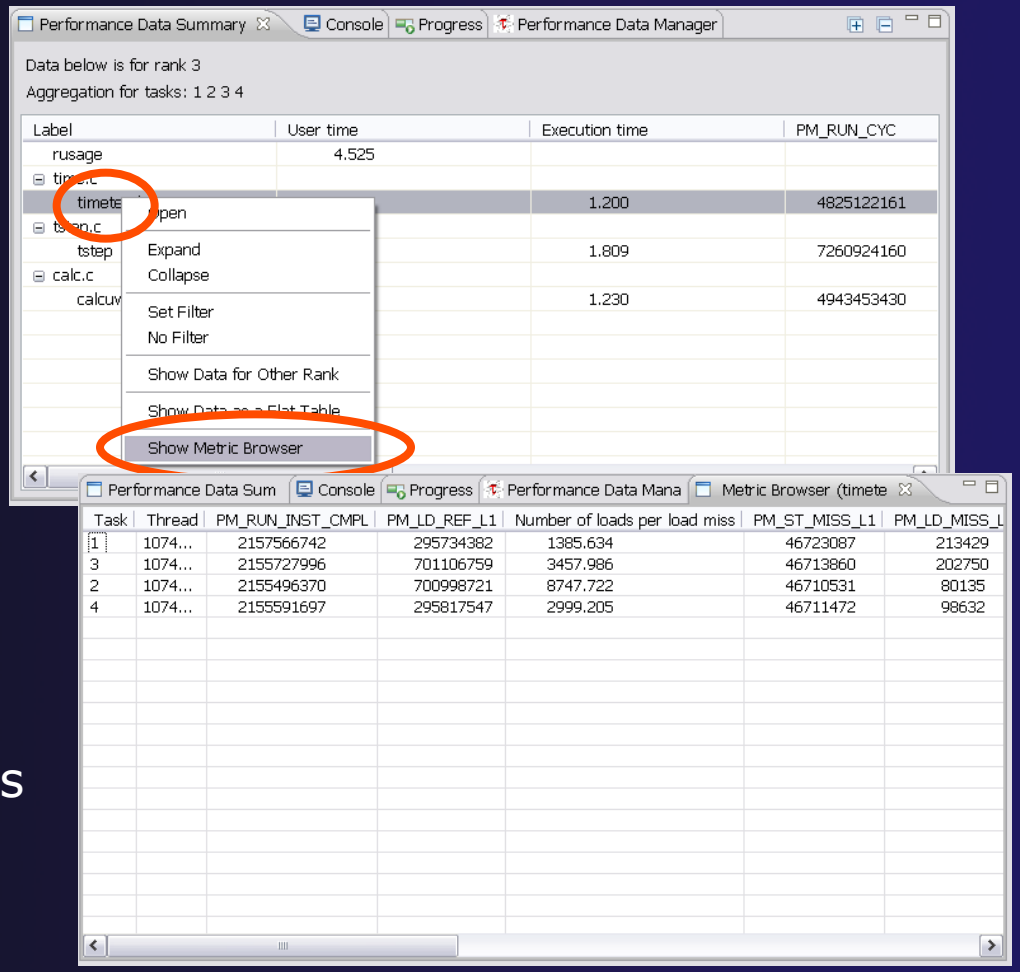
- When viewing multiple tasks
  - Right click on any entry in **Performance Data Summary** view
  - Select **Show Data for Other Rank**
  - Select task/pid from dropdown and click **OK**





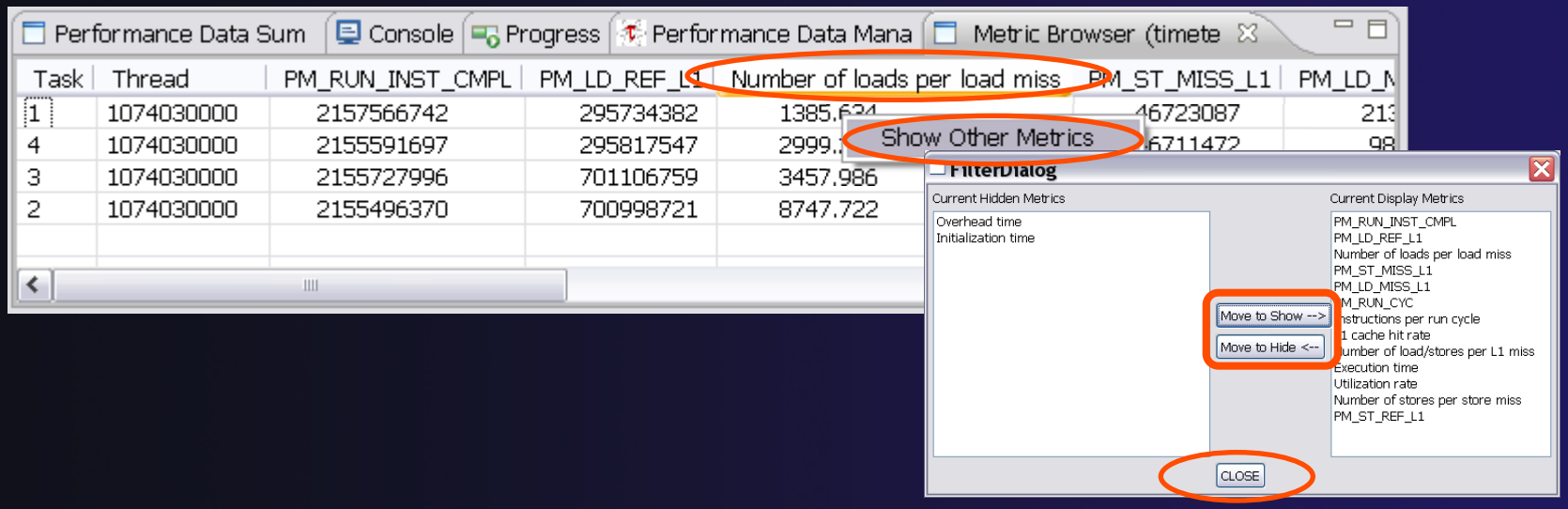
View HPM Performance Data 3

- 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

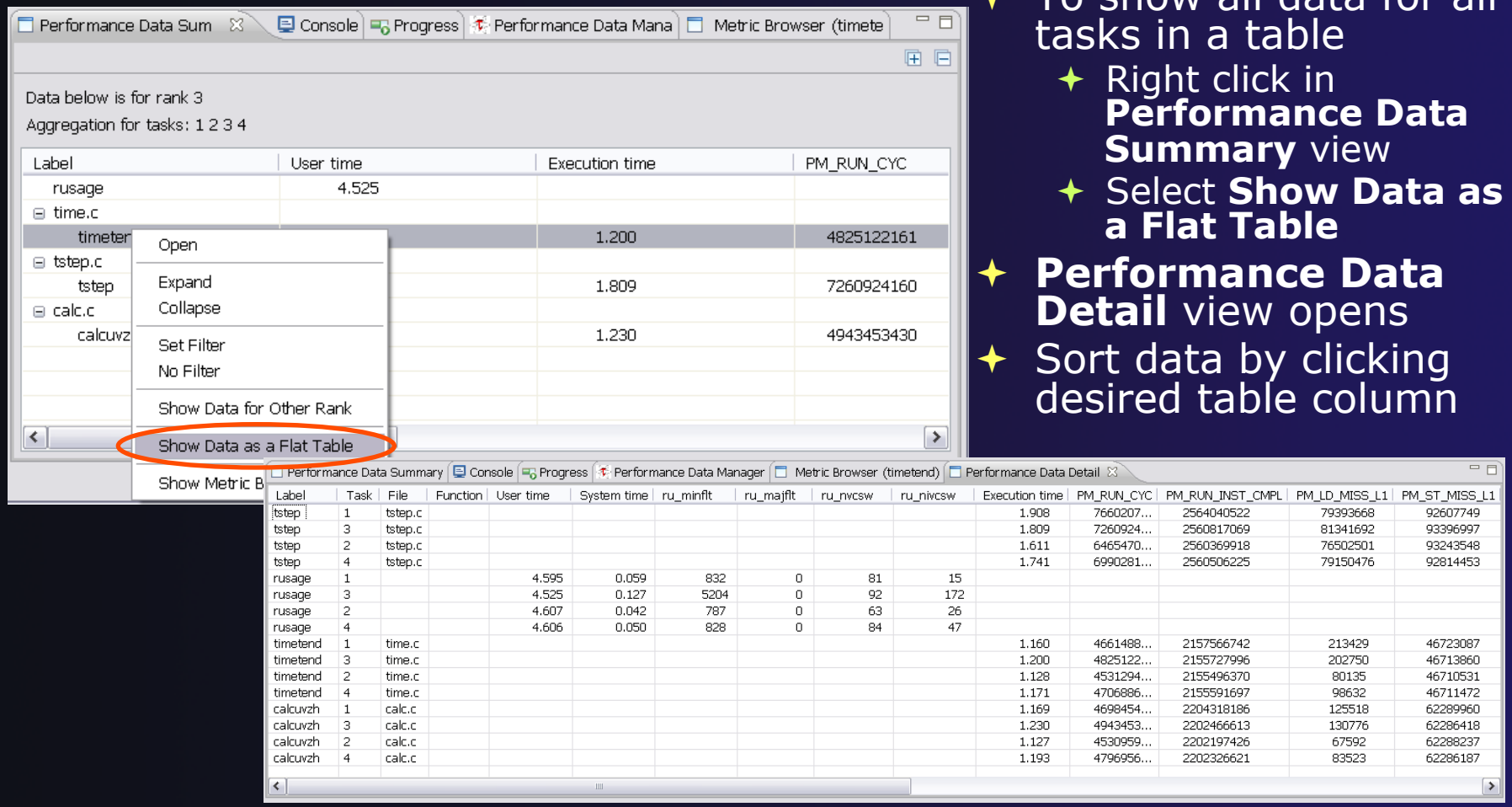


View HPM Performance Data 4

- 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**



View HPM Performance Data 5

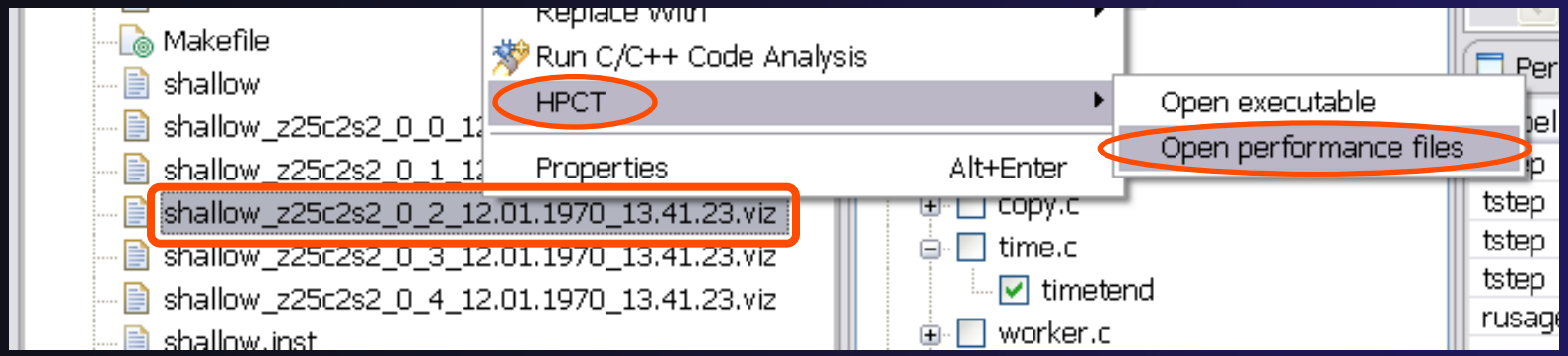


Label	Task	File	Function	User time	System time	ru_minflt	ru_majflt	ru_nvcsw	ru_nivcsw	Execution time	PM_RUN_CYC	PM_RUN_INST_CMPL	PM_LD_MISS_L1	PM_ST_MISS_L1
tstep	1	tstep.c								1.908	7660207...	2564040522	79393668	92607749
tstep	3	tstep.c								1.809	7260924...	2560817069	81341692	93396997
tstep	2	tstep.c								1.611	6465470...	2560369918	76502501	93243548
tstep	4	tstep.c								1.741	6990281...	2560506225	79150476	92814453
rusage	1			4.595	0.059	832	0	81	15					
rusage	3			4.525	0.127	5204	0	92	172					
rusage	2			4.607	0.042	787	0	63	26					
rusage	4			4.606	0.050	828	0	84	47					
timetend	1	time.c								1.160	4661488...	2157566742	213429	46723087
timetend	3	time.c								1.200	4825122...	2155727996	202750	46713860
timetend	2	time.c								1.128	4531294...	2155496370	80135	46710531
timetend	4	time.c								1.171	4706886...	2155591697	98632	46711472
calcuvzh	1	calc.c								1.169	4698454...	2204318186	125518	62289960
calcuvzh	3	calc.c								1.230	4943453...	2202466613	130776	62286418
calcuvzh	2	calc.c								1.127	4530959...	2202197426	67592	62288237
calcuvzh	4	calc.c								1.193	4796956...	2202326621	83523	62286187

- 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



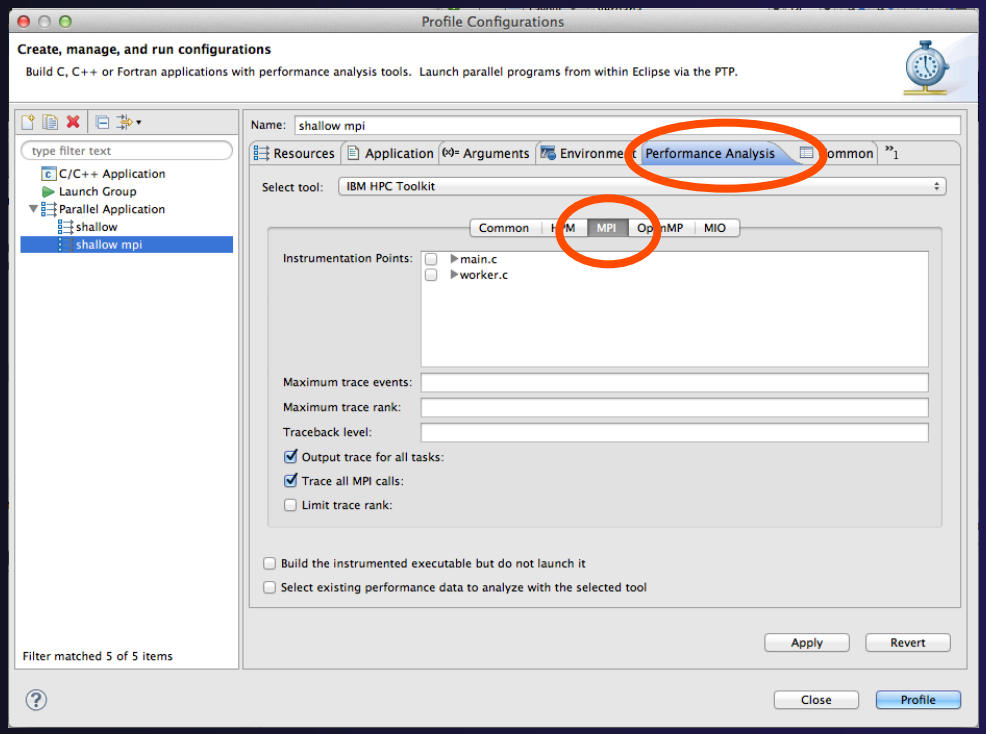
MPI Profiling and Tracing

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

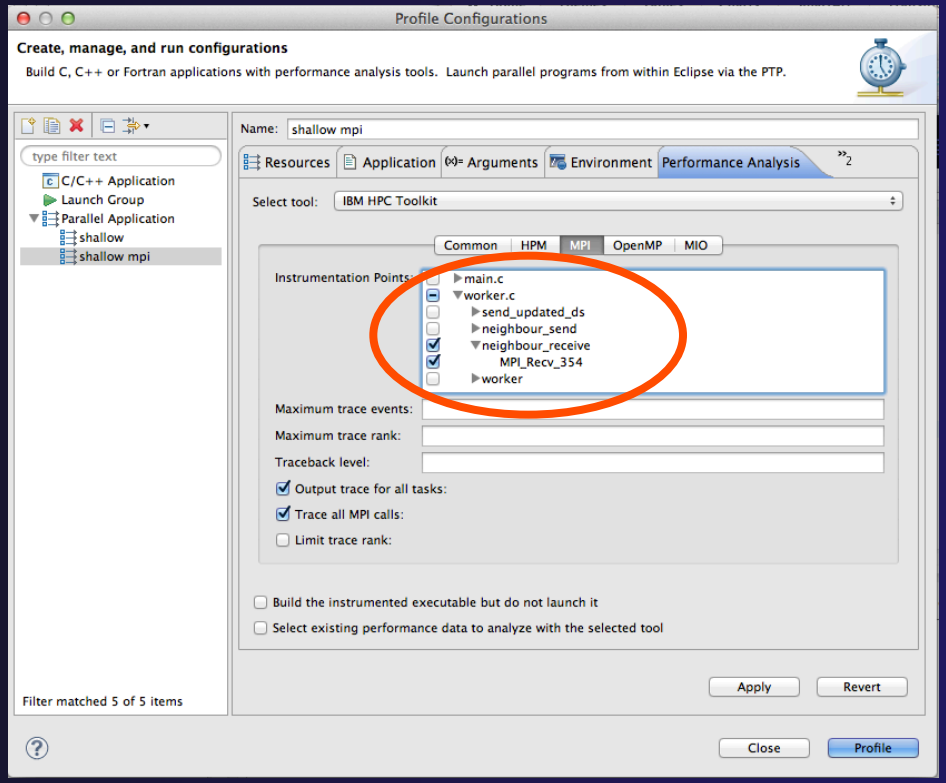
Collect MPI Data

- 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



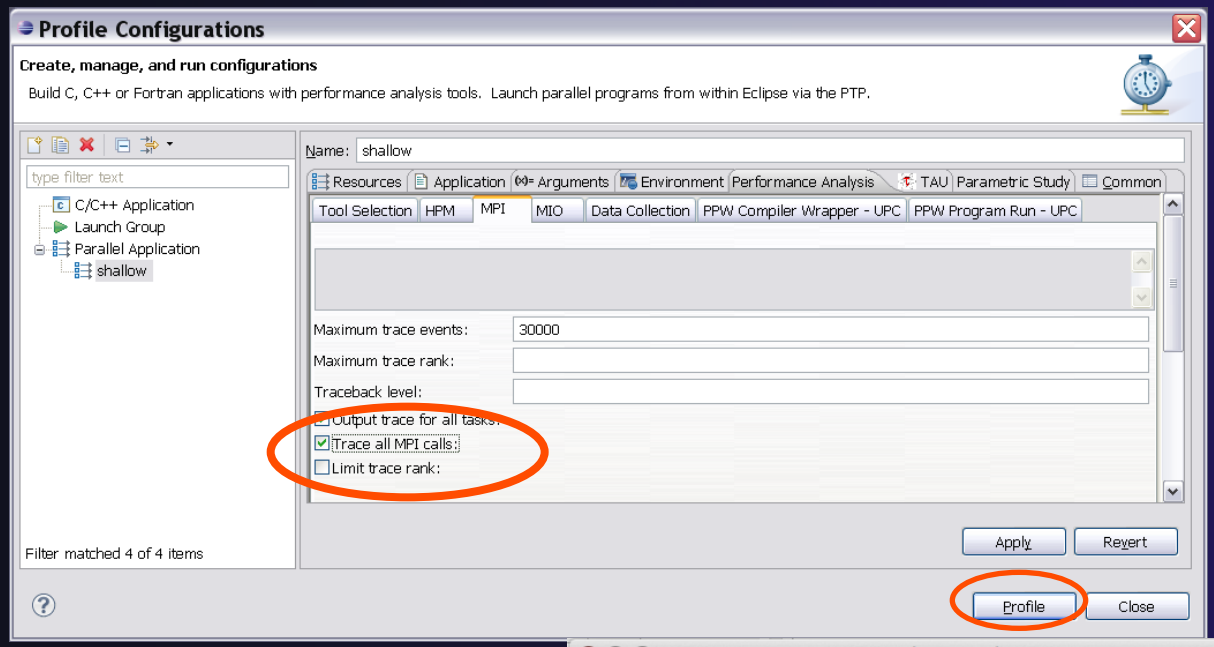
Collect MPI Data (2)

- 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_##

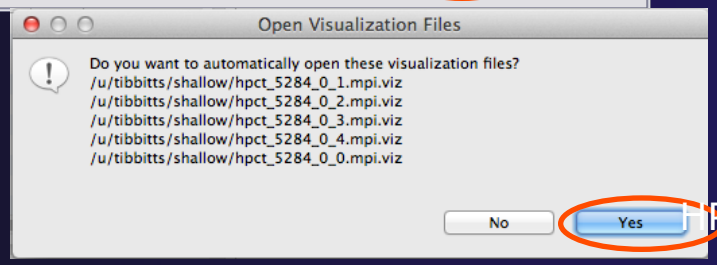


Collect MPI Data 3

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

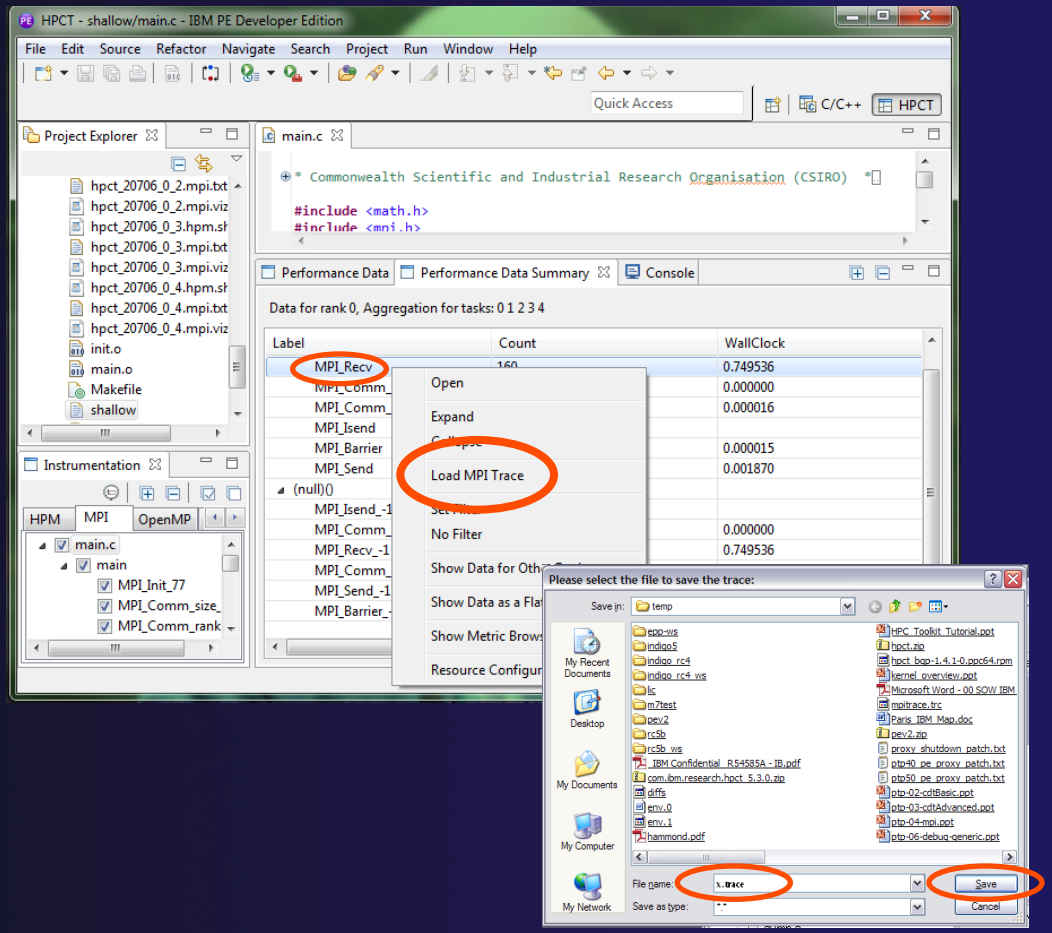


- Select **Profile** to run the profile analysis



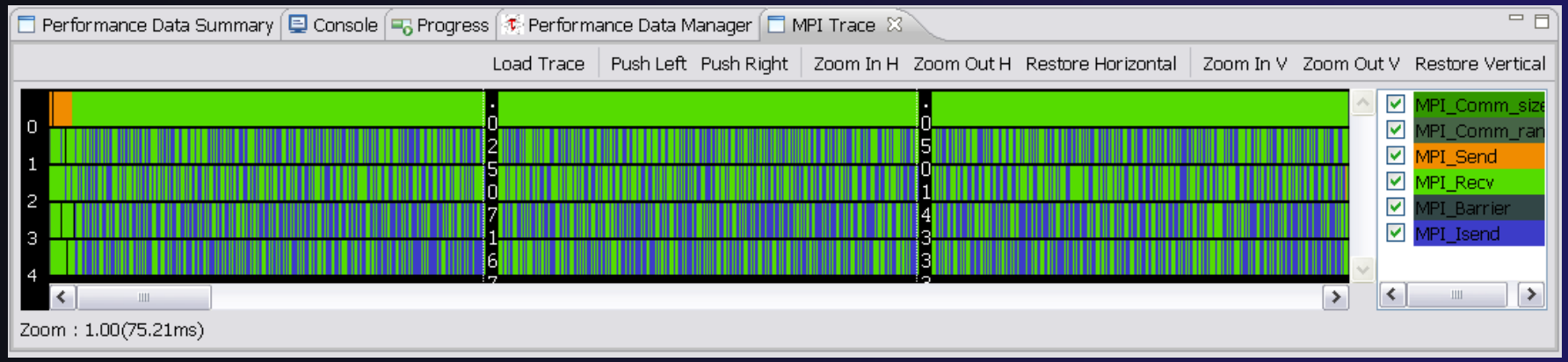
View MPI Profile Data

- **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



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

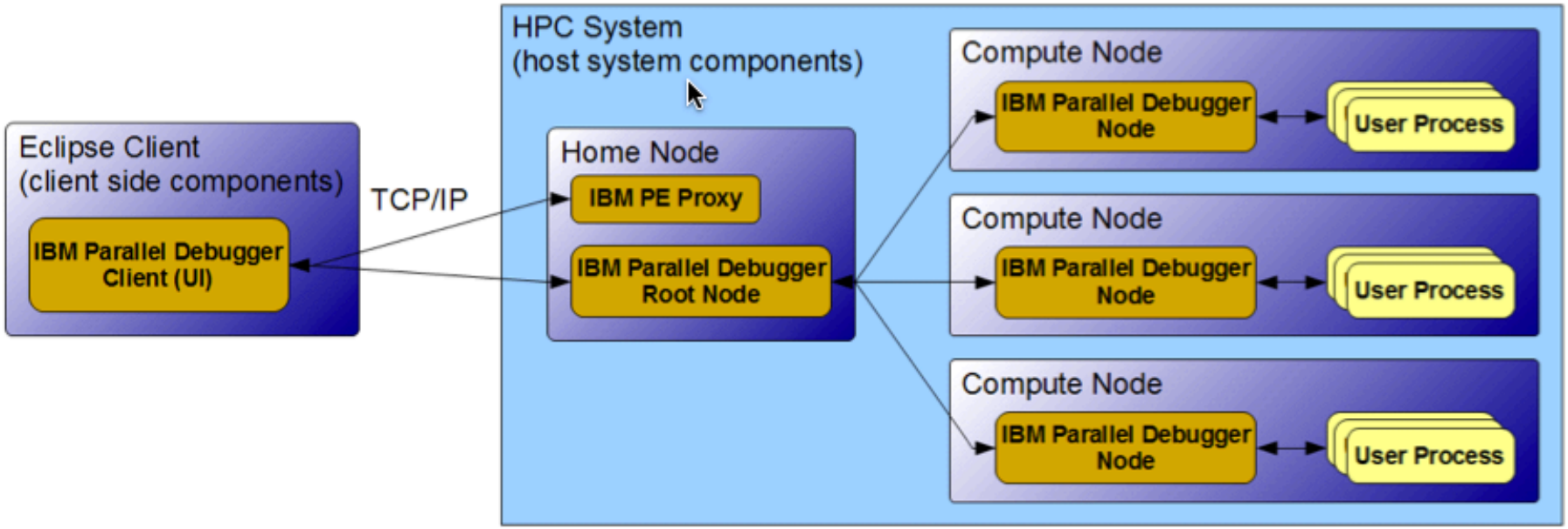
"Debugging code on a parallel machine with hundreds or thousands of cores creates unique problems, and may be the biggest single challenge facing parallel programming"

- Charles Holland [DARPA], The Economist, June 2011

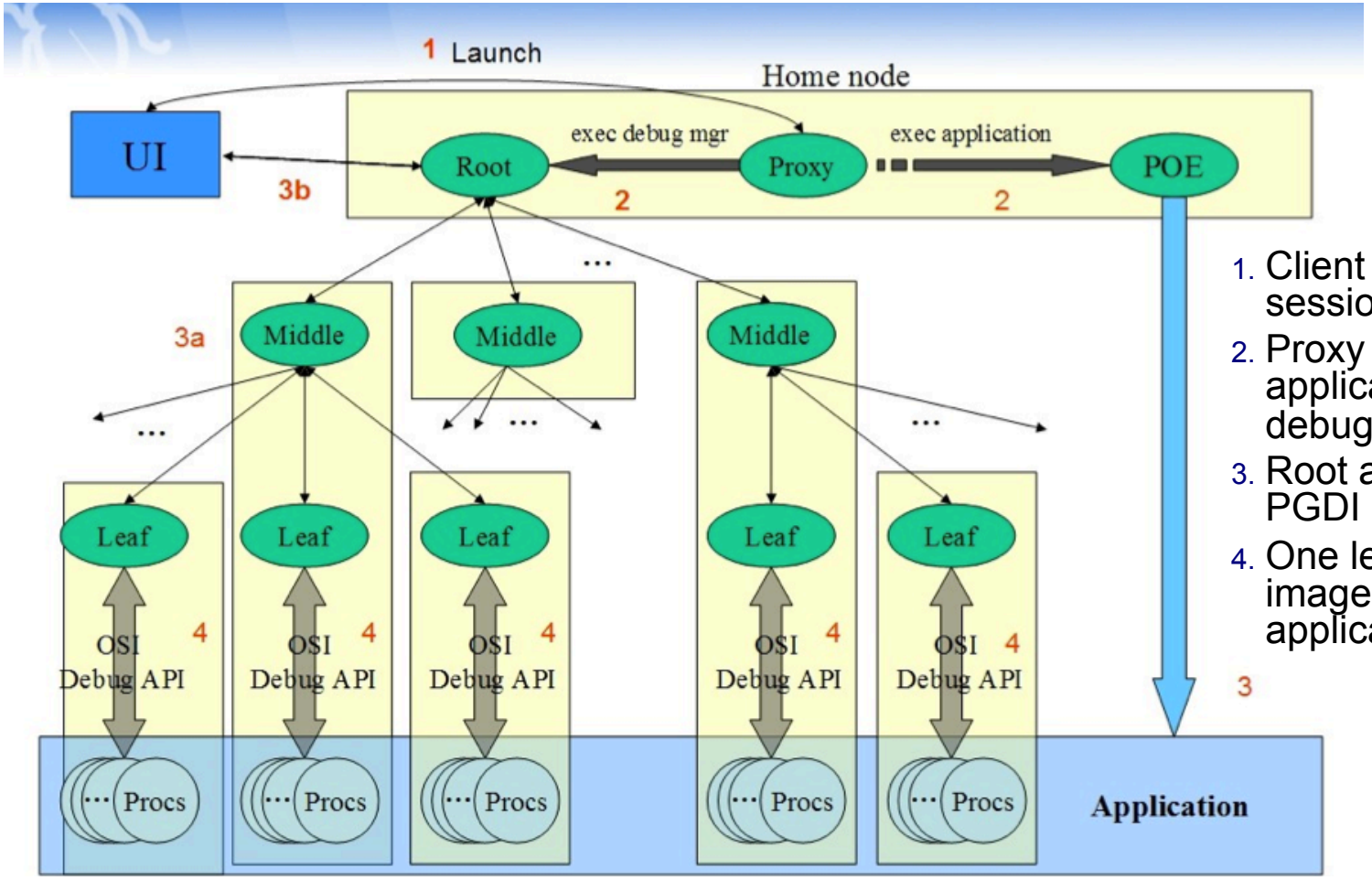
About IBM ParDeb

- 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



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)

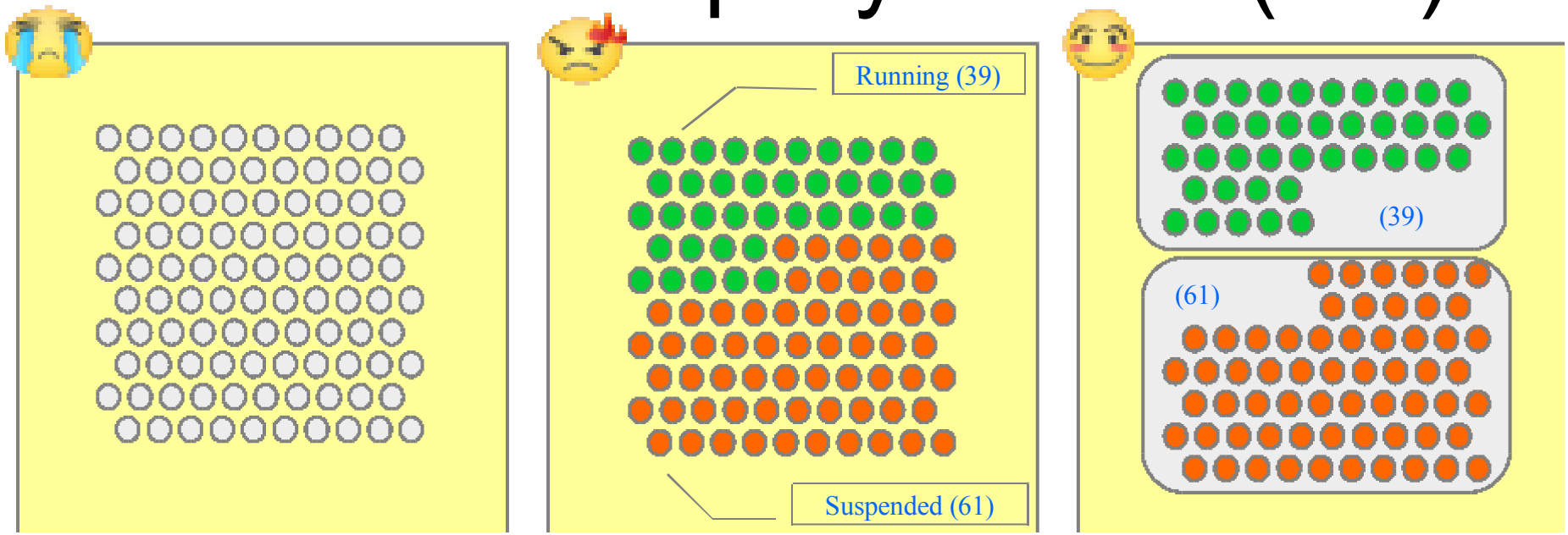
- Organizing the threads into groups
  - Based on common characteristics of thread state and data
  - Can't be done manually
  - What about thread state and data changes?
  - This only works if the debugger does all the heavy lifting

Debug

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

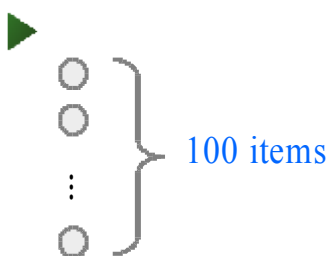
Scalable Group Synthesis (1/4)

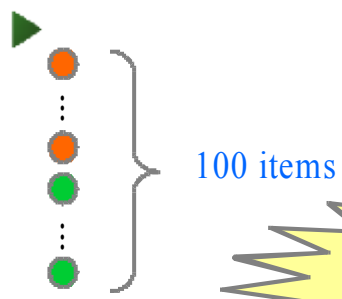


100 threads to manage

2 groups managed manually

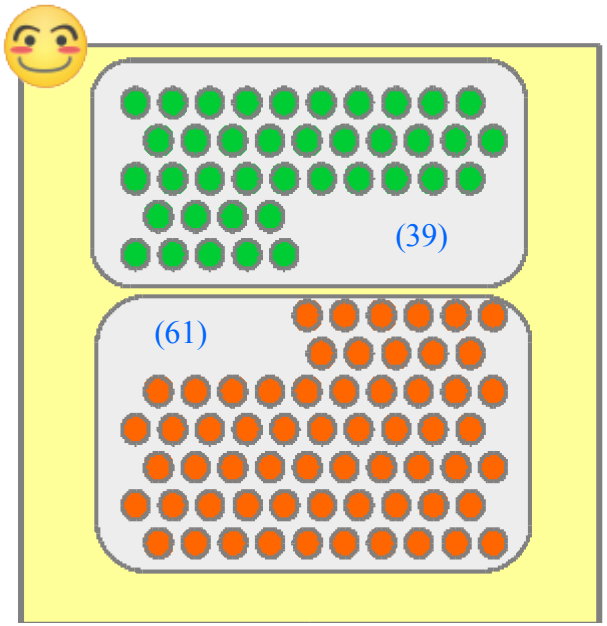
2 synthesized groups (automatically classified)

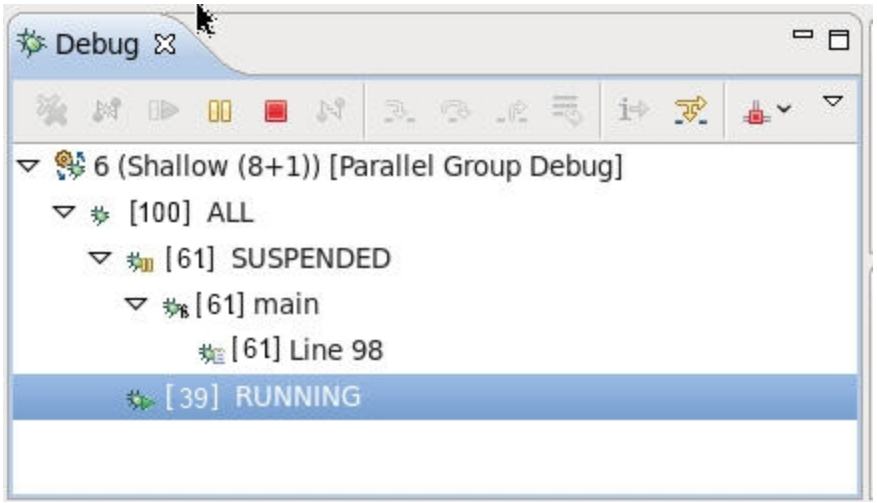




Think about 1M threads!!!

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
  - $Group_{ALL}$ and Suspended => $Group_{Suspended}$
  - $Group_{ALL}$ and Running => $Group_{Running}$
- 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

Debug - Sample Sort/driver.upc - IBM PE Developer Edition

File Edit Navigate Search Project Run Window Help

Debug

4 (Sample Sort) [Parallel Group Debug]
[4] ALL
[4] SUSPENDED
[4] __xlupc_main
Focused Member Session (PID: 71380)
Thread: UPC:2 - xlpgas_tsp_thrd_idler (Runnable)
__xlupc_main : driver.o
main_wrapper : upcrt_setup.o
xlpgas_tsp_thrd_idler : xlpgas_tsp_thrd.o
start_thread : libpthread.so.0:.text

Parallel Debug Groups

Zoom Group

[4] ALL
BREAKPOINTS
STATES

Parallel Stacks

Zoom Group

[4] __xlupc_main (stackDepth=4)
__xlupc_main : driver.upc
main_wrapper : upcrt_setup.c (line 118)
xlpgas_tsp_thrd_idler : xlpgas_tsp_thrd.c (line 109)
start_thread+0x676 : libpthread.so.0

Variables | Breakpoint | Data Distrib | Group Deta | Data Eleme | Parallel Call

Name	Value
A[64]	
bucketSize[4]	
nextThread	0
result[4]	
argc	1
argv	0x00000FFFE4684178

driver.upc

```
75  /* ------------------------------------------------------------------- */
76
77  CLOCK(tstart);
78  switch (codeTuningLevel) {
79    case EXPERIMENTAL: // fall through
80    case TUNED:        init(); break;
81    case SIMPLE:       // fall through
82    default:           init_naive();
83  }
84  CLOCK(tend);
85
86  if (timing && MYTHREAD==0)
87    fprintf(stdout, "Initialize Array: %4.2f\n", (tend - tstart));
88  upc_barrier;
89
90  if (N<=512 && MYTHREAD==0 && 0) {
91    fprintf(stdout, "ARRAY (BF=%d, N=%d):\n  ", BF, N);
```

Outline | Project Explorer

Sample Sort
Scavenger
Shallow

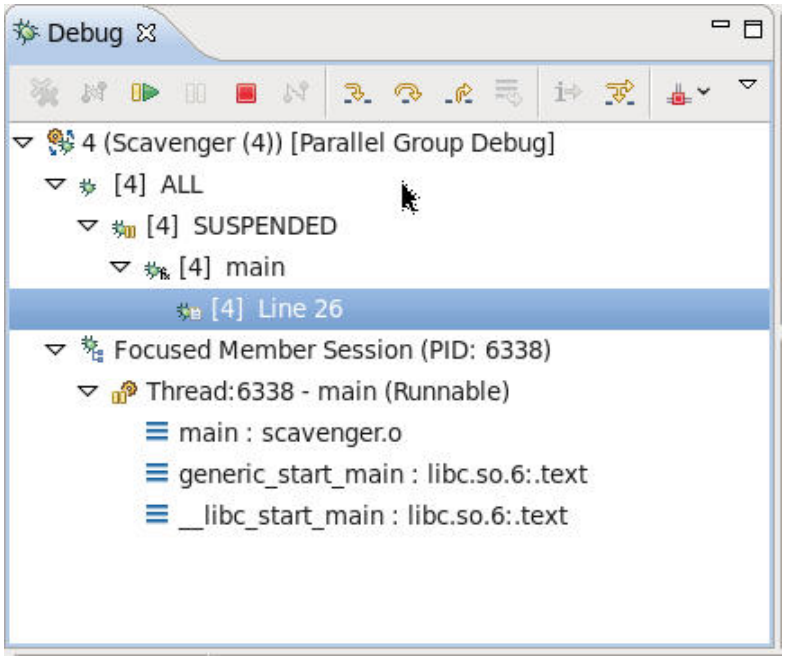
Console | Tasks | Problems | Executables

Sample Sort [Parallel Group Debug] IBM PE@Workshop P7IH System: job_4

```
Verification of result is enabled.

Debug Enabled. Debug Level 2.
```

Remote Tools DStore ...ystem): (100%)

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

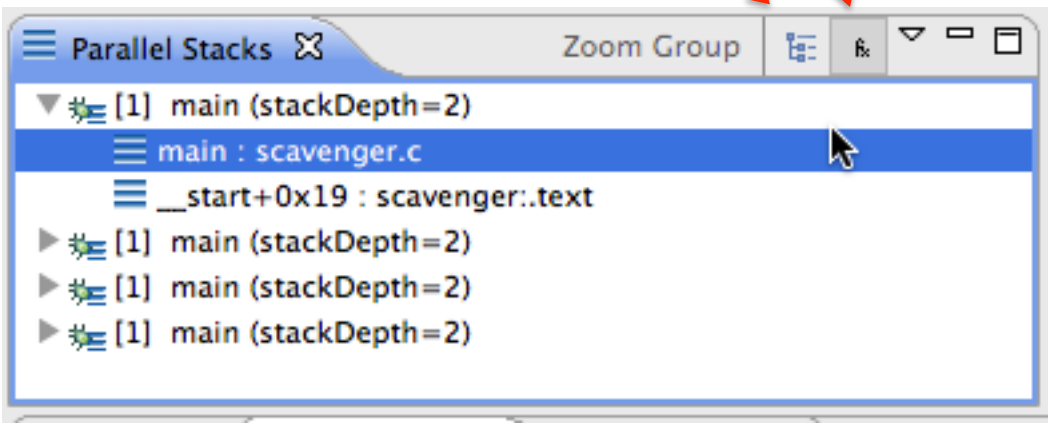
scavenger.c

```
36      long age;
37
38      something = doSomeMoreMagic();
39
40      switch(child)
41          {
42          case 0: /* execute unit of code 1 */
43              num = foo();
44              num = goo(4);
45              break;
46          case 1: /* execute unit of code 2 */
47              num1 = foo_other_file('l');
48              ally = (char) num1;
49              age = light_other_file(53);
50              break;
51          case 2: /* execute unit of code 2 */
52              foo_header_file('b');
53              getThis = doSomeMagic();
54              break;
55          case 3: /* execute unit of code 2 */
56              abc = nowWhat();
57              break;
58          default: /* execute default action */
59              break;
```

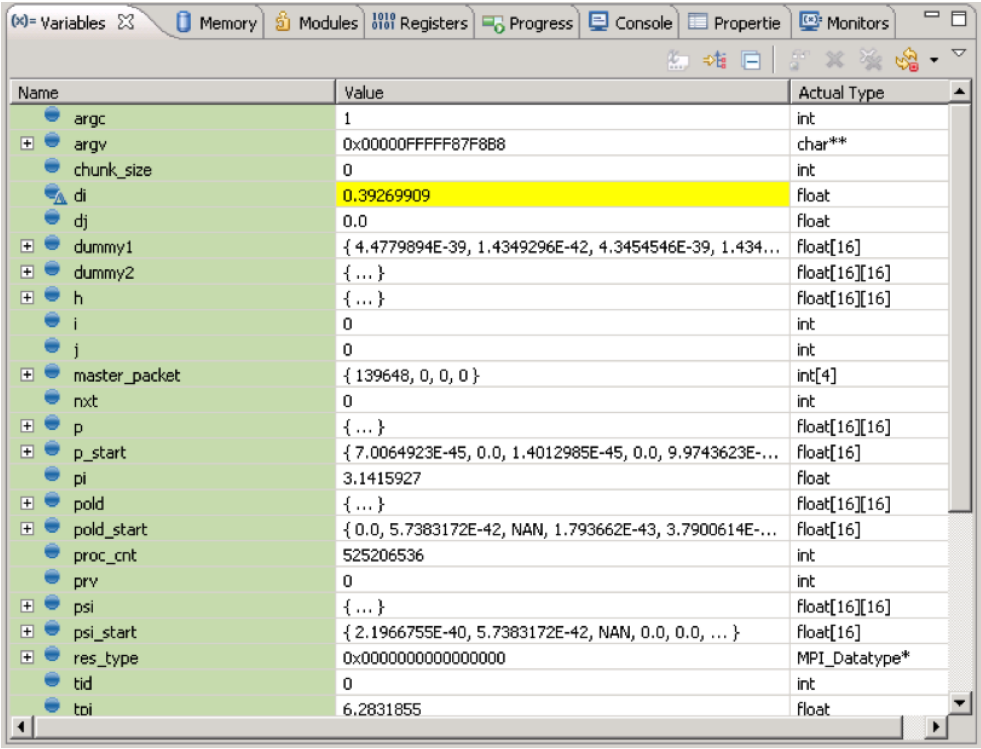
- Synchronized with debug view
- Left Margin
  - Breakpoint markers
  - Execution context markers
- Right Margin
  - File level execution markers
- Line highlighting for suspended threads
  - Orange shading
    - Active thread(s) selected in Debug view
  - Blue dashed box
    - Active thread(s) not selected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

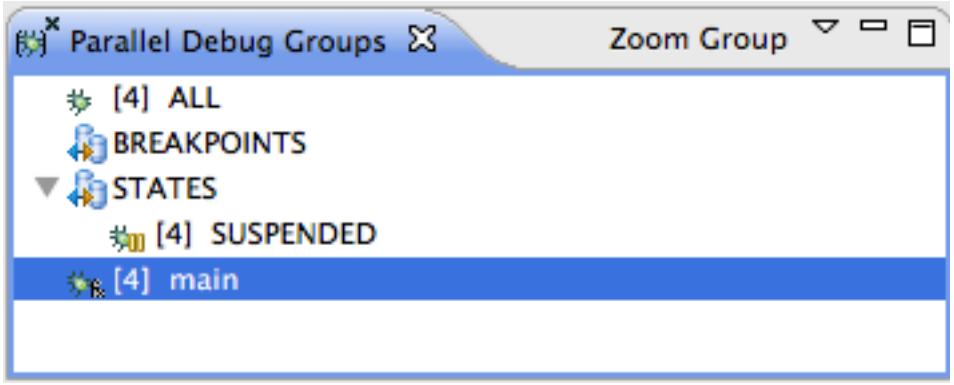


Name	Value	Actual Type
argc	1	int
argv	0x00000FFFFF87F8B8	char**
chunk_size	0	int
di	0.39269909	float
dj	0.0	float
dummy1	{ 4.4779894E-39, 1.4349296E-42, 4.3454546E-39, 1.434...	float[16]
dummy2	{ ... }	float[16][16]
h	{ ... }	float[16][16]
i	0	int
j	0	int
master_packet	{ 139648, 0, 0, 0 }	int[4]
nxt	0	int
p	{ ... }	float[16][16]
p_start	{ 7.0064923E-45, 0.0, 1.4012985E-45, 0.0, 9.9743623E-...	float[16]
pi	3.1415927	float
pold	{ ... }	float[16][16]
pold_start	{ 0.0, 5.7383172E-42, NAN, 1.793662E-43, 3.7900614E-...	float[16]
proc_cnt	525206536	int
prv	0	int
psi	{ ... }	float[16][16]
psi_start	{ 2.1966755E-40, 5.7383172E-42, NAN, 0.0, 0.0, ... }	float[16]
res_type	0x0000000000000000	MPI_Datatype*
tid	0	int
tpi	6.2831855	float

- 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

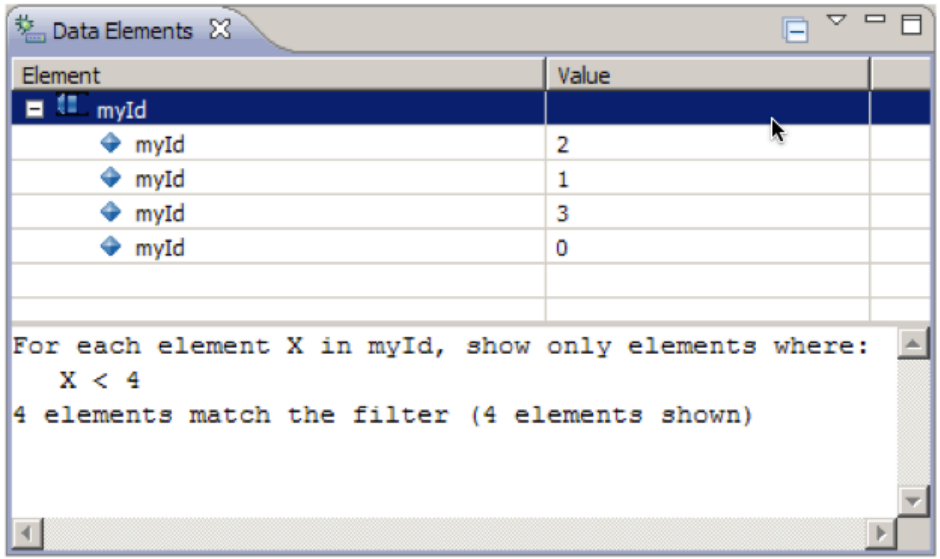
Parallel Debug Groups View

- 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



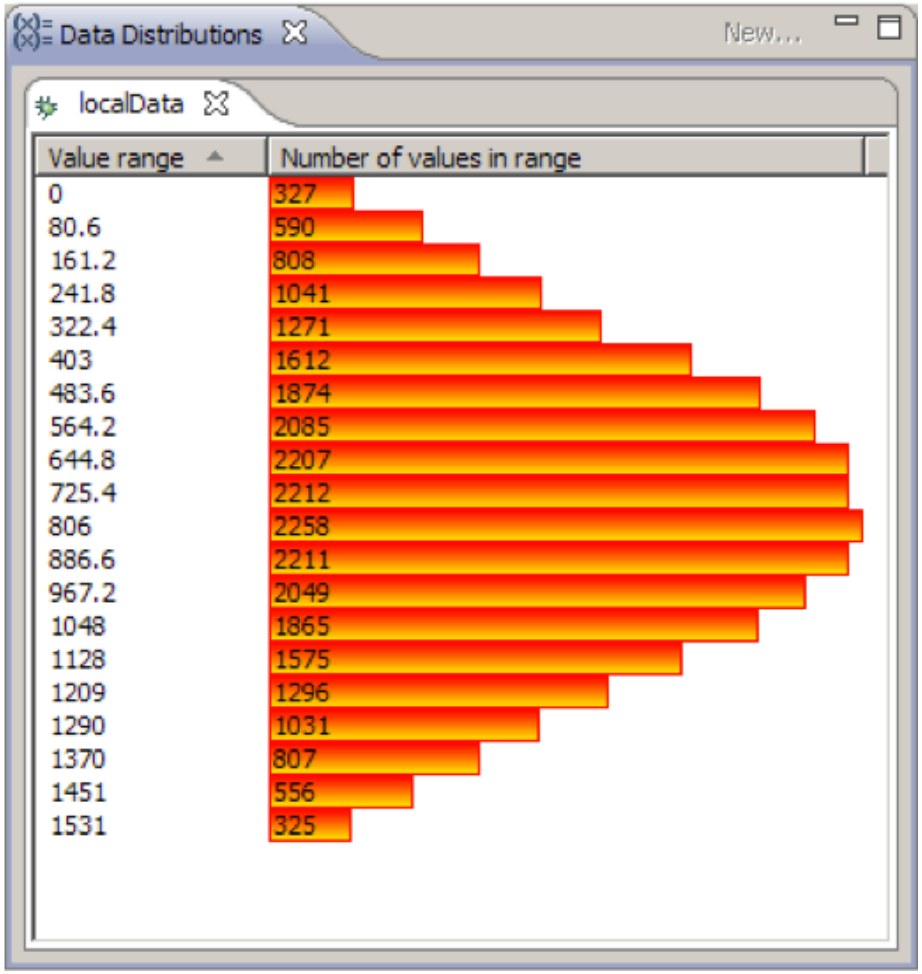
Data Elements View

- 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

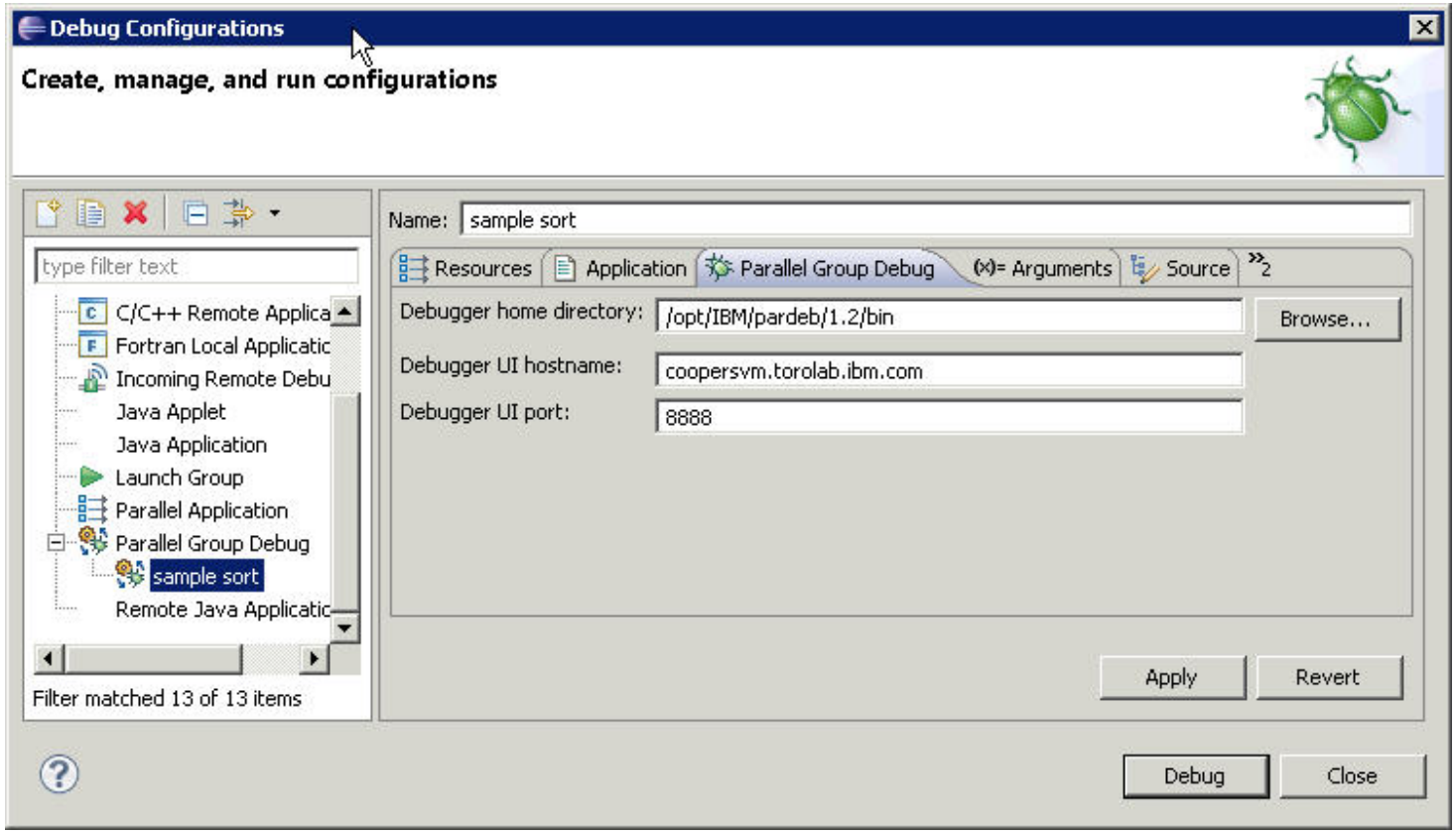


Data Distributions View

- 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

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

PTP Tutorial Feedback

- Please complete feedback form
- Your feedback is valuable!

Thanks for attending
We hope you found it useful