Introduction to Eclipse/ICE and Eclipse Parallel Tools Platform

Jay Alameda, NCSA
jalameda@ncsa.illinois.edu

Slides by
Jay Jay Billings, Greg Watson, Beth Tibbitts,
Jay Alameda, Galen Arnold, Steve Brandt, Chris Navarro, Jeff Overbey, and Wyatt Spear

August 18, 2015

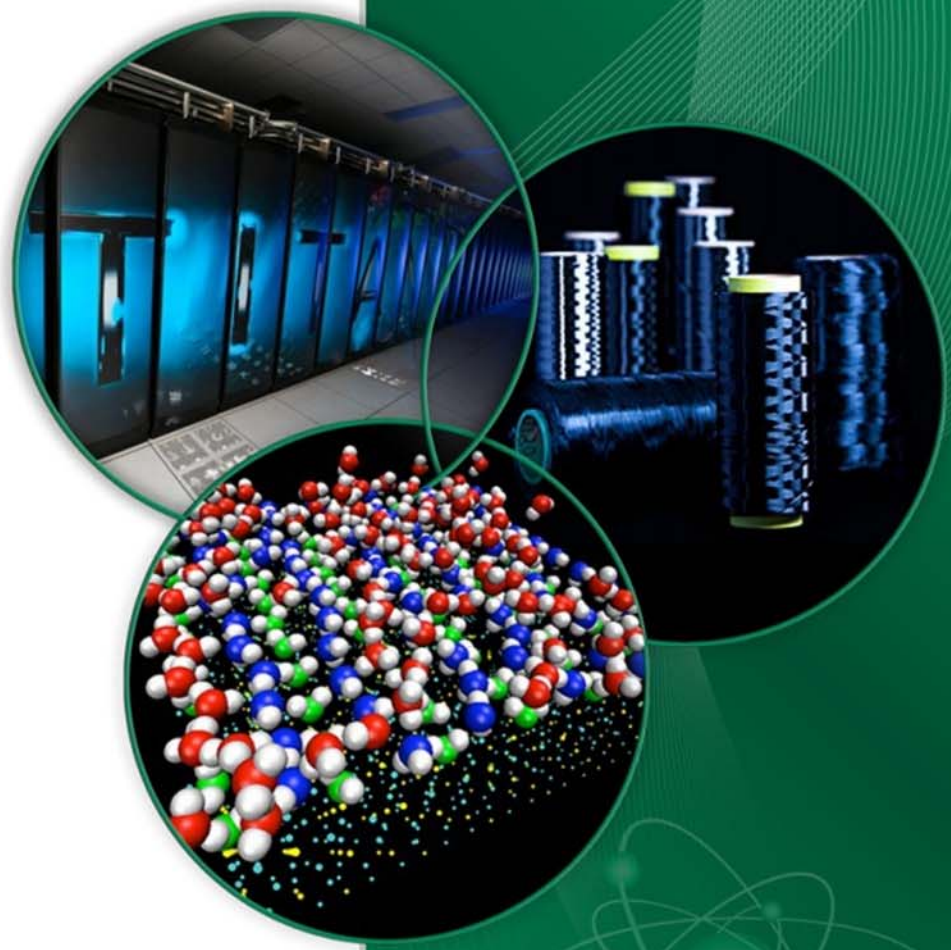
Portions of this material are supported by or based upon work supported by
- The Defense Advanced Research Projects Agency (DARPA) under its Agreement No. HR0011-07-9-0002
- The Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (award number OCI 07-25070)
- The United States Department of Energy under Contract No. DE-FG02-06ER25752
- The SI2-SSI Productive and Accessible Development Workbench for HPC Applications, which is supported by the National Science Foundation under award number OCI 1047956

Outline

Time (Tentative)	Module	Topics	Presenter
2:00-2:30	Eclipse/ICE Intro/Overview	✦ Overview of Eclipse/ICE	Jay
2:30-3:00	Eclipse PTP Basics	✦ Eclipse Overview ✦ Synchronized projects ✦ Git support ✦ Building with Make ✦ Target system configurations ✦ Launching a parallel application ✦ Modules/environment mgmt	Jay

Installation instructions (and these slides) are available at
http://wiki.eclipse.org/PTP/tutorials/XSEDE15

Eclipse ICE: ORNL's Modeling and Simulation User Environment



Jay Jay Billings
Research Staff
Oak Ridge National Laboratory
The Bredesen Center
billingsjj@ornl.gov
@jayjaybillings
+jayjaybillings

with the rest of ICE Team

20141212

Outline

- A little about usability
- The problem
- Our solution
- Applications

Additional Resources: http://www.eclipse.org/ice

Sourceforge.net

niceproject.sourceforge.net

YouTube

youtube.com/jayjaybillings

Ohloh.net

ohloh.net/p/niceproject

Usability

Usability

Usability → Accessibility

How can users get access to advanced M&S capabilities?

Think of using a highly technical code as the bear trap at right:

- Easily getting the good stuff requires training
- Features may exist in a rusty framework with sharp edges (tough code, not always bad)
- Adding something is not for the faint of heart

Making it easier to use a code *naturally* exposes advanced features, masks complex designs and enables users to explore!

Usability in Modeling and Simulation

How I send messages around the World:

How I send messages to my code:

./xolotl ../benchmarks/he-W_2067.txt --handlers dummy --petsc -ts_final_time 1000 -ts_final_time 1000 -ts_adapt_dt_max 10 -ts_max_snes_failures 200 -pc_type fieldsplit -pc_fieldsplit_detect_coupling -fieldsplit_0_pc_type redundant -fieldsplit_1_pc_type sor -ksp_monitor -ts_max_steps 3

Really?!

So what's the problem?

Standard Model of Scientific Computing

All users must do these things...

Define the Problem

Write an input file in a format reminiscent of a dead language

Run the Simulator

Manually launch jobs on impressively terrifying machines

Analyze Output

01100010
01101001
01101110
01100001
01110010
01111001

Analyze simulation output in its most raw and unlimited form

Archive Output

Store data... somewhere!

Super-users think these are easy tasks, but most users are overwhelmed!

But that doesn't cover all M&S workflows!

It more or less does...

Define the Problem

Write an input file in a format reminiscent of a dead language

1 1..n

Run the Simulator

Manually launch jobs on impressively terrifying machines

2 n+1 1

Analyze Output

01100010
01101001
01101110
01100001
01110010
01111001

Analyze simulation output in its most raw and unlimited form

3 n+2 2

Archive Output

Store data... somewhere!

4 n+3

These are activities that are mixed and matched to produce an outcome.

Standard — Single Use — Multi-Input — "The Re-Run"

A cooler model of Scientific Computing

It would be better to have a computer program handle all of that...

A. User

Define the Problem

Run the Simulator

Analyze Output

01100010
01101001
01101110
01100001
01110010
01111001

Archive Output

Most of the stuff we need to do can be encapsulated for ease of use and/or automated entirely with improvements.

A cooler model of Scientific Computing

"Most of the stuff we need to do can be encapsulated for ease of use and/or automated entirely with improvements."

So why not do it?

The Eclipse Integrated Computational Environment (ICE)

Or, the product also known as the NEAMS Integrated Computational Environment

And in some places as “Jade”

Project Details

- 100% open source, cross-platform, general-purpose user environment
- Primarily developed at Oak Ridge National Laboratory
- 7 team members, (6 full time, 1 part), + 2 support staff
- Focused on making life easier for users *and* developers

Define the Problem

Run the Simulator

Analyze Output

01100010
01101001
01101110
01100001
01110010
01111001

Archive Output

ICE does these three very well

Needs work

Code details of the Project

- Binary and source downloads available
  - Don't burden users with compiling!
- Roughly 200k lines of code
  - +200k comments
  - Extensive (exhaustive!) tests
  - All Eclipse based (well, mostly)
- Good out-of-source documentation
  - Wiki with tutorials, detailed docs
  - UML Model
  - Javadoc & Doxygen
- Sponsored by U.S. DOE, LDRD, WFO
- Will be released through the Eclipse Foundation in FY15

Languages

Java	74%	C++	17%
XML	7%	10 Other	2%

Project Cost

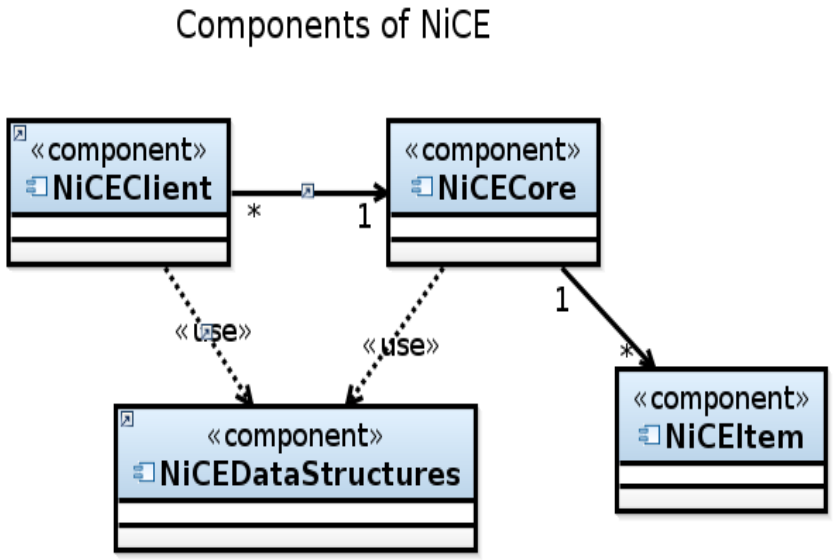
Codebase Size
141,172 lines

Estimated Effort
35 person-years

Estimated Cost
$ 1,943,471 *

*Using the Basic COCOMO Model

How does it work?



The platform is easily modified:

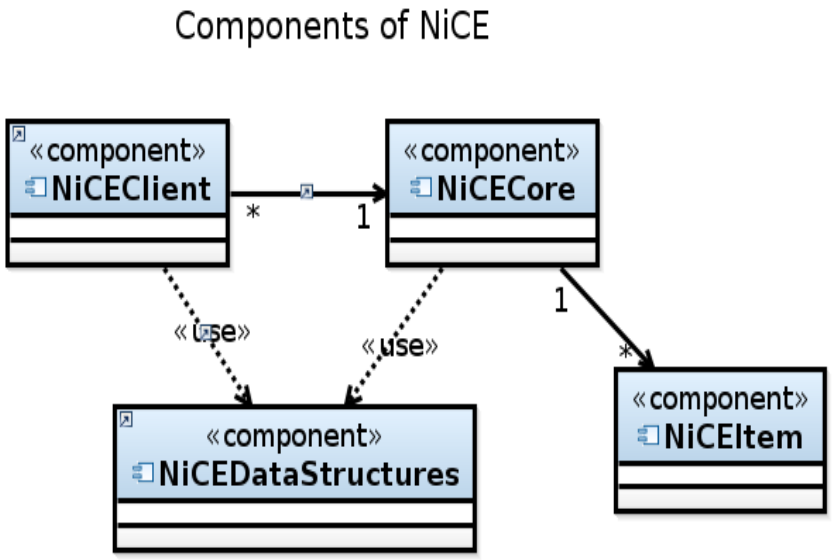
- Developers write *plugins*
- Some extensions can be made with no code (no plugins)

Plugin code goes here!

Users...

- Download a binary for their OS on our website
- Unzip a file to install for most capability, some dependencies for nuclear users
- Follow online tutorials or join our mailing list for help
- Otherwise, point and click!

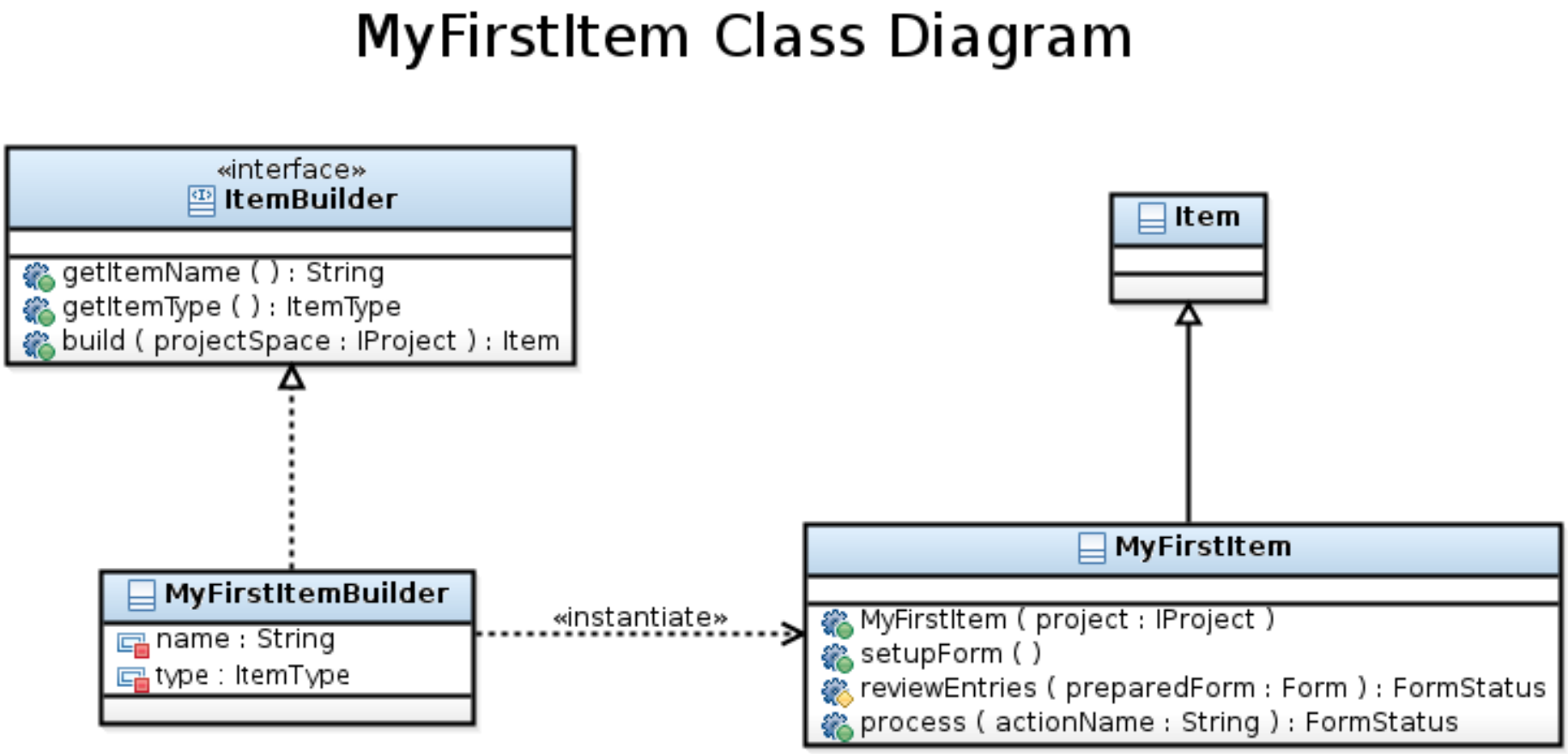
How does it work? Part 2



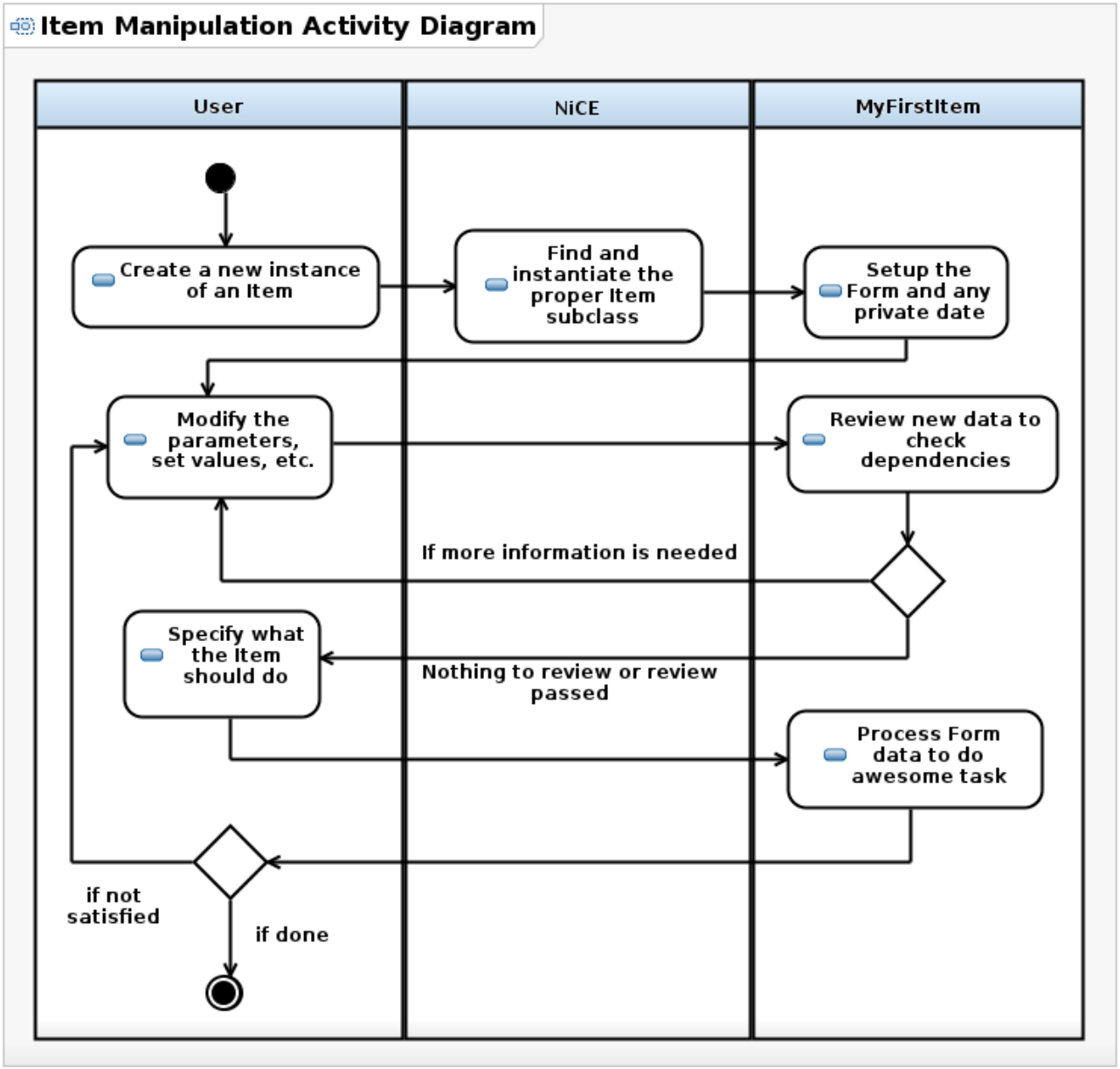
Plugins are:

- Dynamic Services - Completely reusable components!
- “Item” Subclasses - Most of the work is already done by the platform
- Self-contained, business logic - **ONLY** your code, not UI, etc.
- Tools - Reusable components, tools, or things other

How does it work? Part 3



How does it work? Part 4



How does it work? Part 5

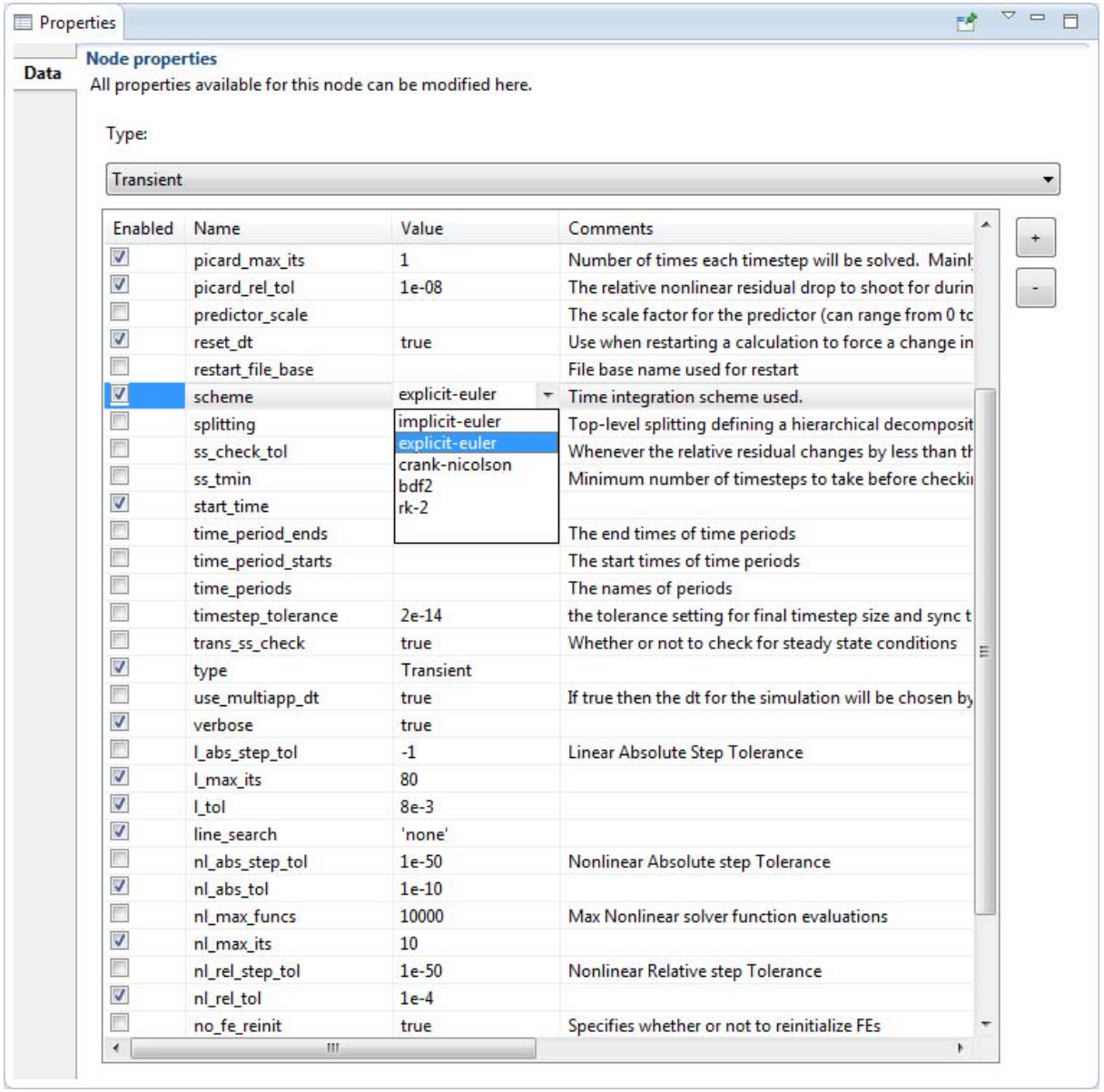
Things to keep in mind:

- You only write business code
- UI and marshalling are provided by the platform (unless you want to extend it)
- Codify only what is needed; reuse what you already have (preprocessors, etc.)

How does it work? Part 6

All of the data structures are backed by sophisticated tools so you deal with your domain.

Standardization for the win!

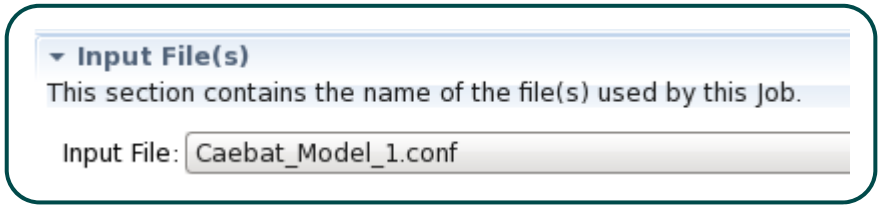


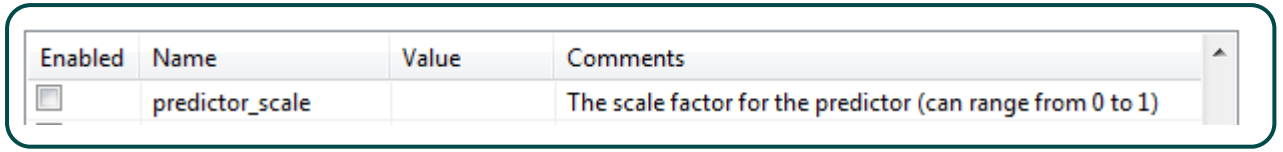
Different views of the same data

```
<Entry defaultValue="1.7899" ready="true" changeState="false">
    <AllowedValueType>Undefined</AllowedValueType>
</Entry>
```

```
entry1 = new Entry() {
        protected void setup() {
            allowedValues = new ArrayList<String>();
            allowedValues.add("0");
            allowedValues.add("50");
            defaultValue = "1";
            allowedValueType = AllowedValueType.Continuous;
        }
};
entry1.setName("Generic 1");
```

All of these are logically equivalent because of the standardization!





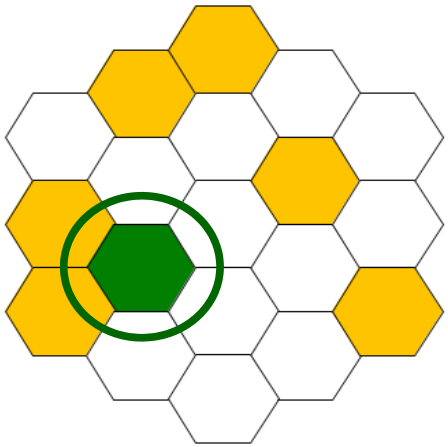
“How about a magic trick?”

Remember: A picture is worth 1000 words

Also Remember: Everything you see is 100% reusable

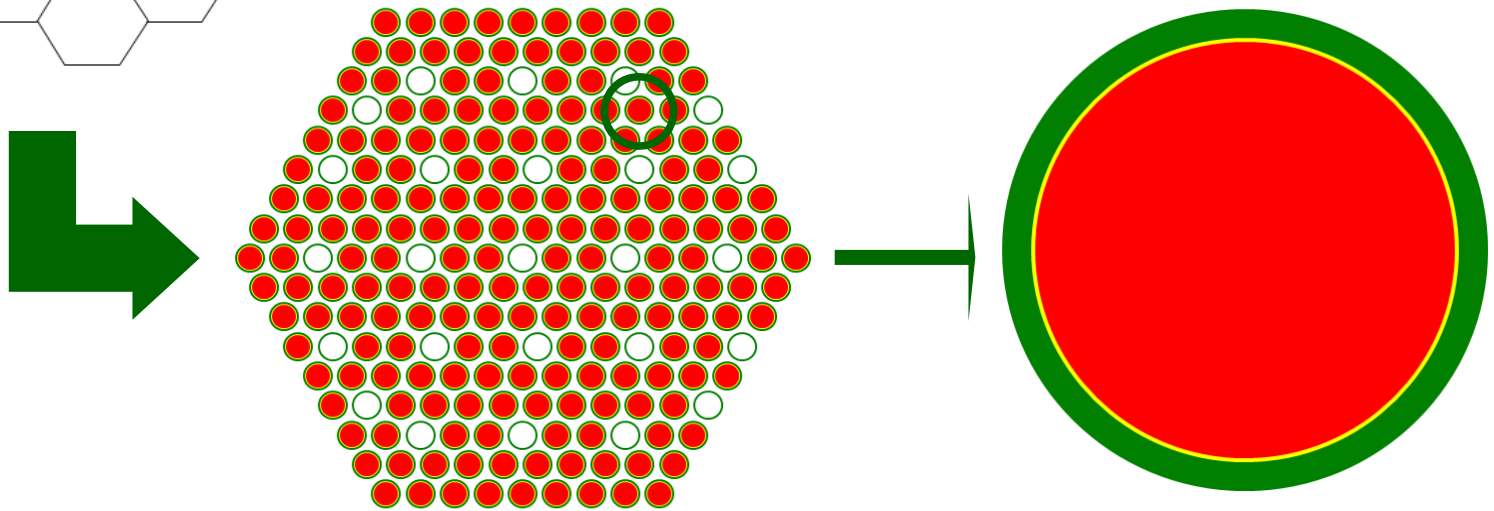
Domain-specific views

Raw data can be messy. What if we let the user explore the reactor to find the data they want?



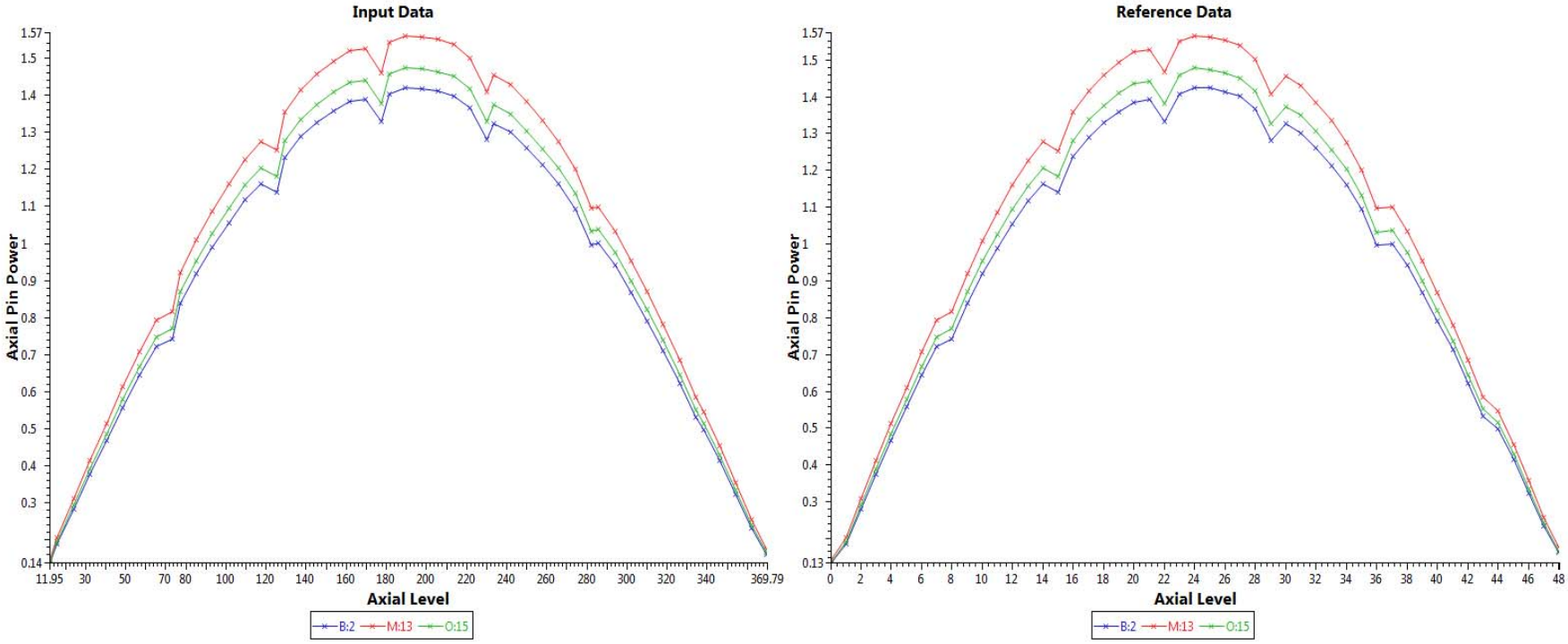
Mimic the hierarchical layout of real reactors:

- Cores have assemblies
- Assemblies have rods or pins
- Rods have exotic materials



Enable Visual Comparative Analysis...

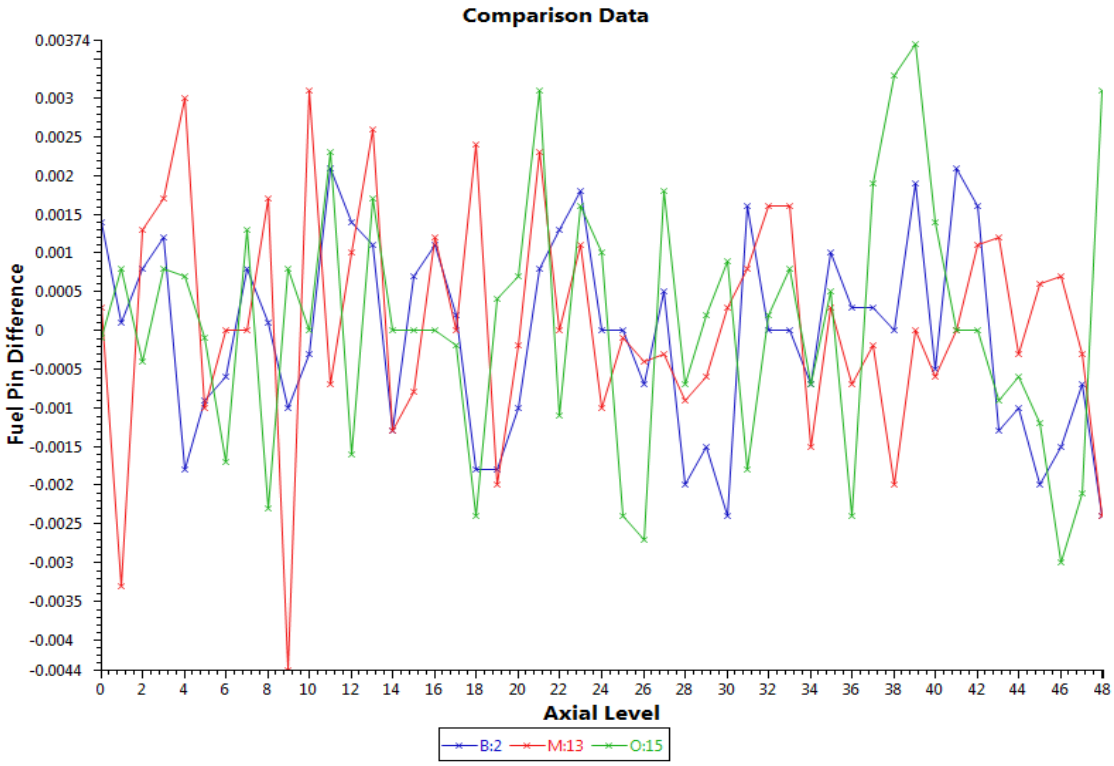
Goal 1: Examine output results by visually comparing them with validated reference data.



Input fuel pin powers along the length of the pin for three pins on the left, with reference data for the same pins on the right

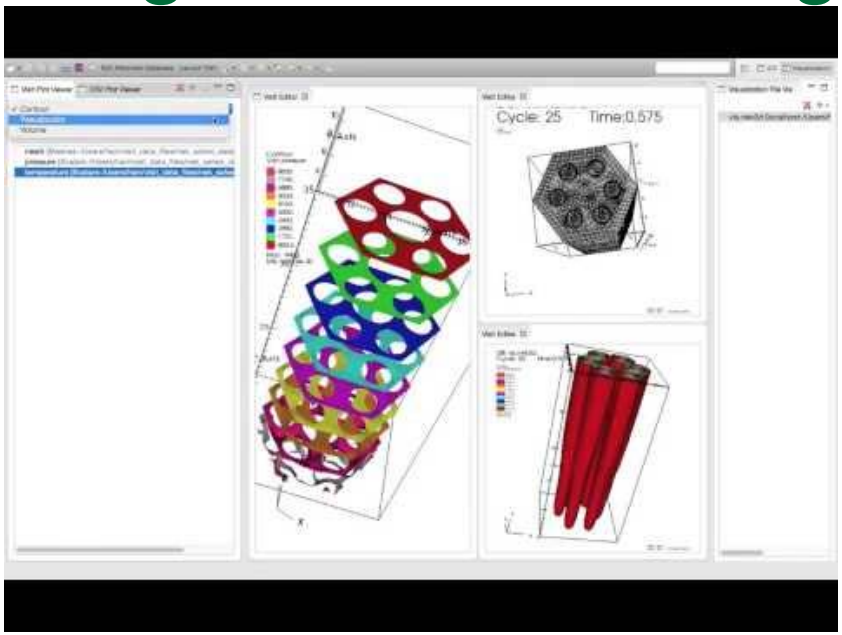
... and Quantitative Comparative Analysis

Goal 2: Do the actual math, too, to quantitatively compare results based on both the simulation data and the reference data.

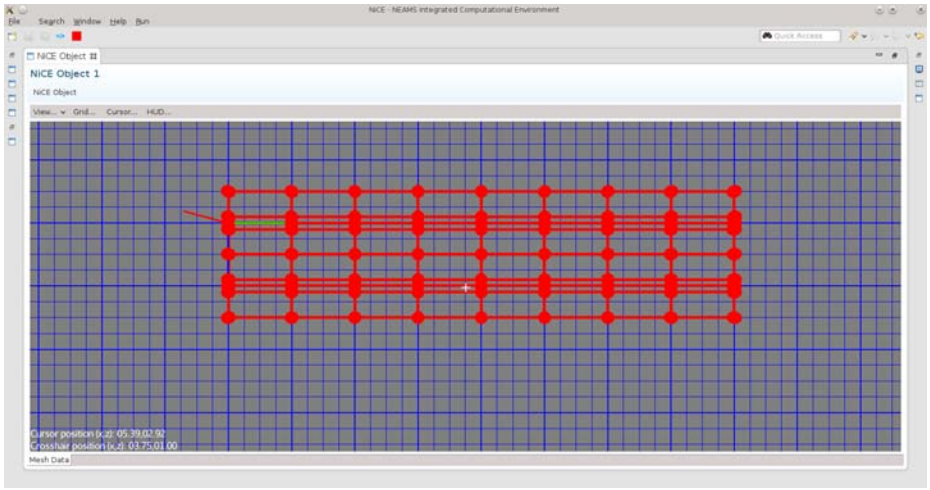


The percent difference between the simulation output and reference power data for the pins.

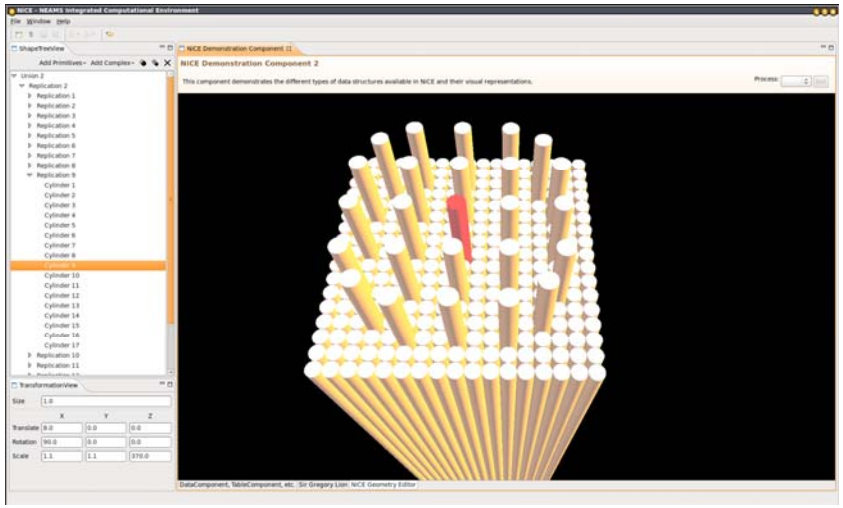
Integrate with existing tools to get a leg up

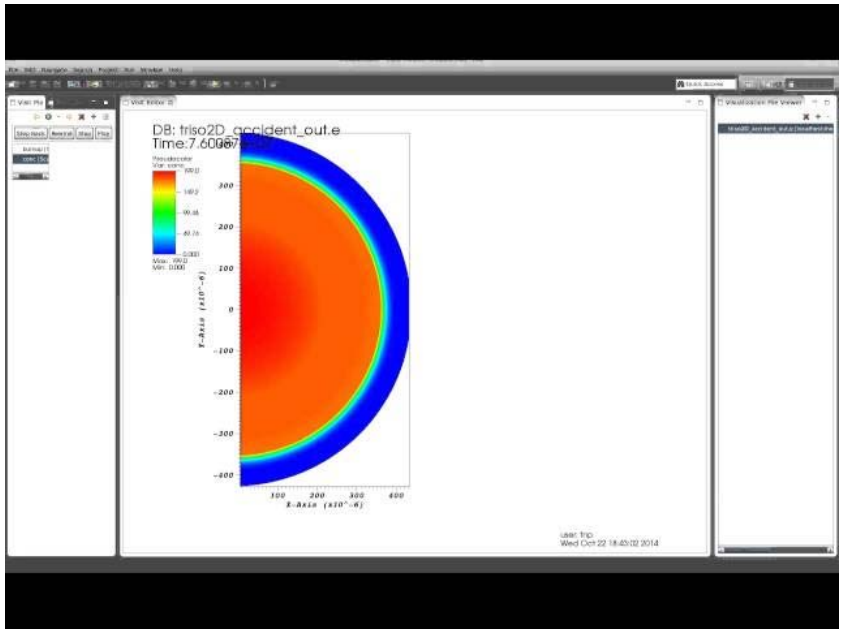


Fully interactive 3D support in VisIt

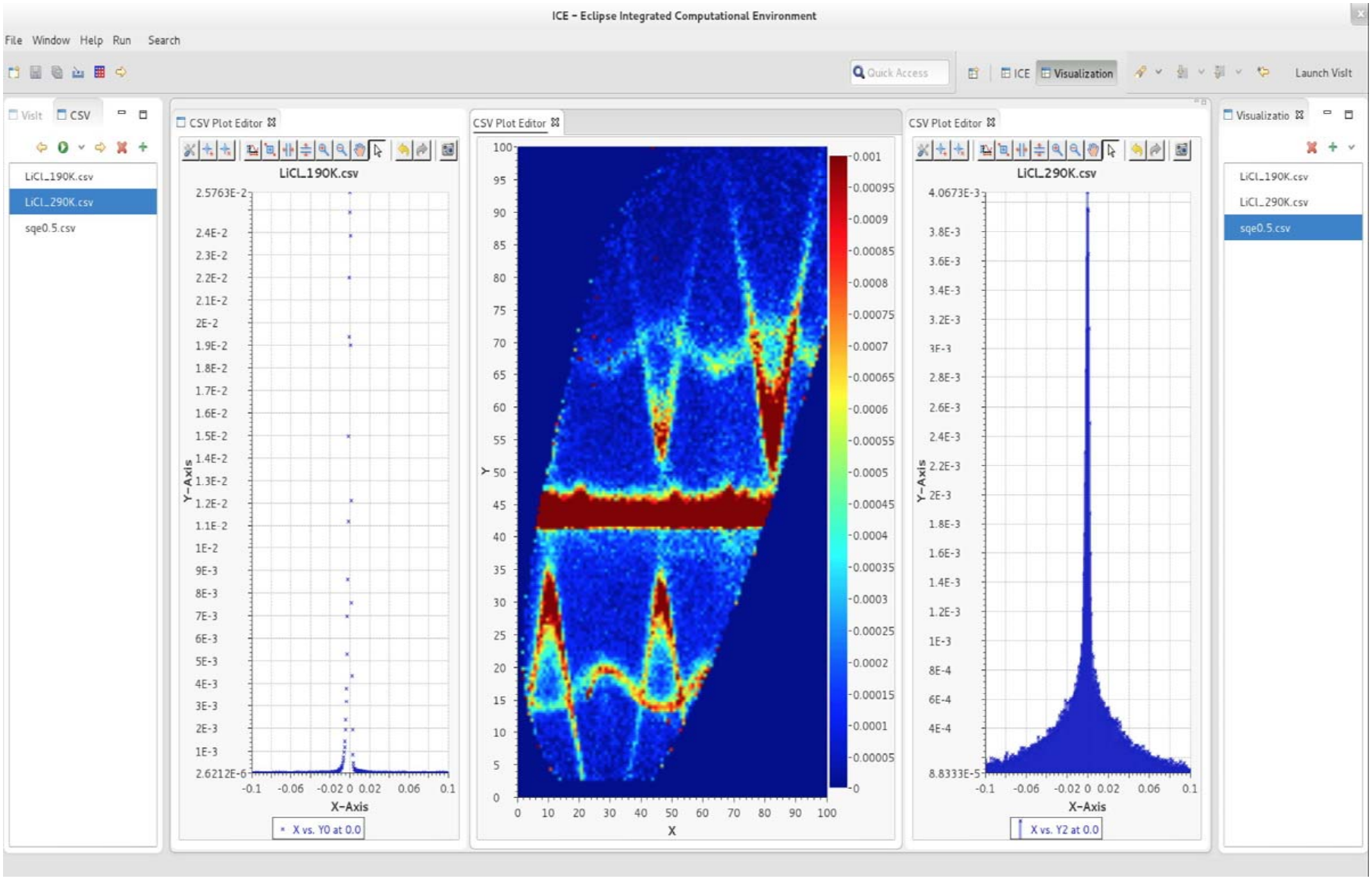


JMonkeyEngine for Mesh and Geometry

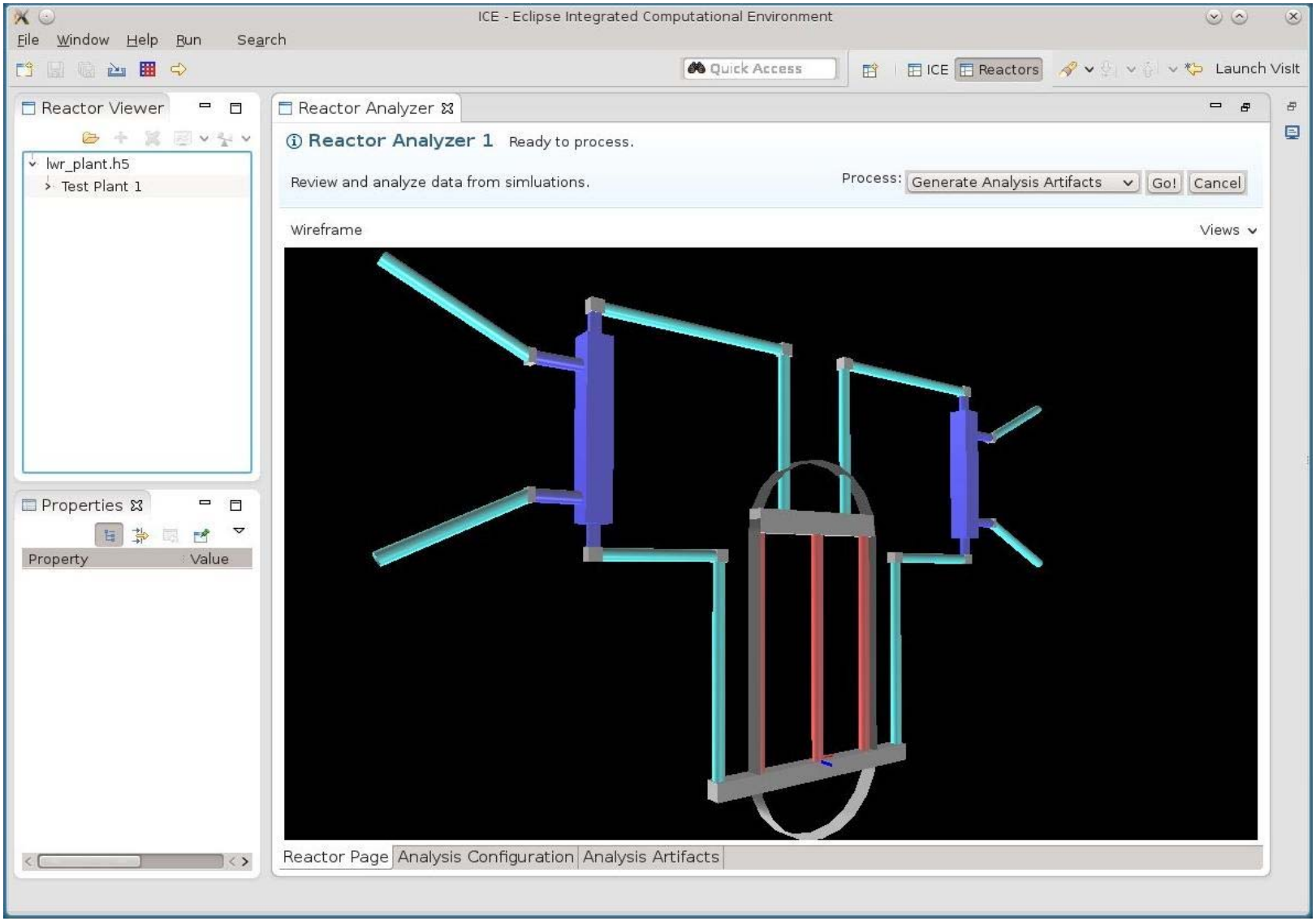




An early video of the triso results in ICE's visualization perspective (thus the bugs!)



SNS Phonon Scattering Data



3D Model of a Nuclear Plant (TMI)

Where does it work?

Nuclear Energy

Data Analysis

Batteries

Quantum Computing

Basic 3D Geometry and 2D Mesh Editing

Fusion

Advanced Materials

Astrophysics

Climate

Buildings

More 3rd Party Tools

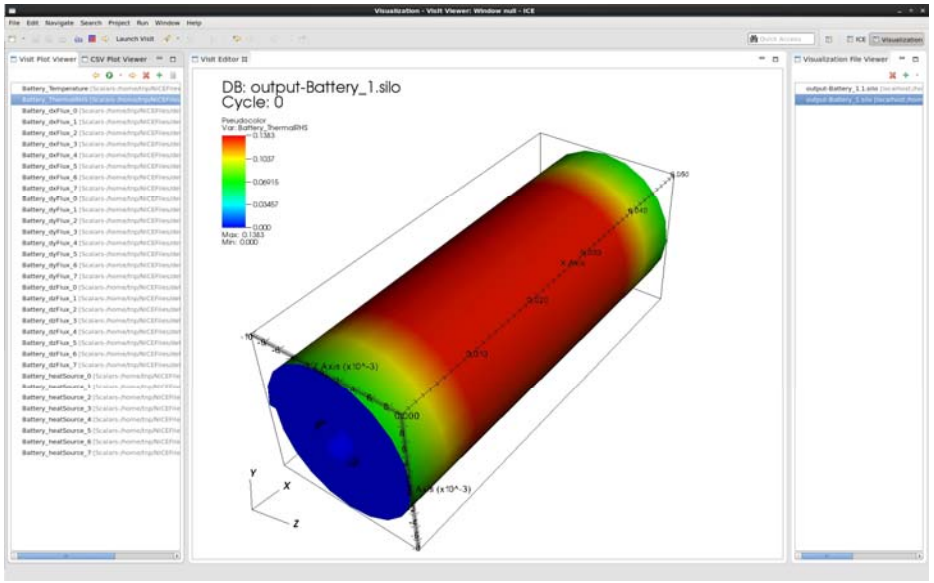
Coming in FY15!

Where does it work?

Supported Projects

- Batteries - CAEBAT & BatML
- MOOSE Apps - Bison, Marmot, Relap- 7
- Quantum Computing - JADE
- Advanced Materials - Sassena, Reflectivity
- Viz & Data - VisIt
- Viz & Data - General Purpose Plotting
- Viz & Data - Reactor Analyzer
- Viz & Data - 3D Constructive Solid Geometry
- CFD - Nek5000
- Viz & Data - Mesh Editor
- Nuclear Energy - PROTEUS

One platform; N-integrated codes



Other CS things happening that I didn't have time to discuss...

- Web server (Officially! We've had it unofficially for awhile.)
- Project/Repo support (Git, SVN, etc.)
- Paraview integration
- Provenance tracking
- 3D mesh editor
- Automated (declarative) data mining
- Auto-generation of plugins from XML
- Developer-focused productivity plugins
- Collaboration-focused extensions
- Cloud… stuff
- Automatic updates

Among 100 other different things...

The Eclipse Foundation

The Eclipse Foundation is...

- a community for developing commercially-friendly open source software
- responsible for developing the massive Eclipse Platform
- a nonprofit, member supported organization
- currently hosting over 250 projects (NiCE was #251!)
- a “main line” into collaborations with high-profile organizations
- blessed with nine million users worldwide
- essentially the NFL of software development

But what does it all mean for ORNL? Why did we do this?

The Eclipse Foundation, p2

Lots of value for ORNL:

- License issues checked and code released under industry friendly license
- Rigorous software development community helps improve our tech
- Common build infrastructure, etc.
- ICE is exposed to a very large number of users at conferences, in articles, and as part of releases.
- Streamlines contributions coming *BACK* to ICE

How we now take contributions...

Collaboration Outlook

Core Developers

ORNL
LBNL
UTK

Subject Matter Experts

ANL
INL
UT-Dallas

Contributors

Diamond Light Source
Duquesne University

Hopeful Future Contributors

DOE IT
Kitware Inc.

The NiCE team is multi-institutional and growing!

Our goal is to foster an open, collaborative community that includes contributions of all forms from everyone and encourages constructive, positive discourse about and the implementation of highly usable open source modeling and simulation tools with an license that supports wide adoption of the platform.

We hope to double the number of core developers and core developer institutions in the next calendar year.

Collaboration Opportunities

We want YOU to join our team!

Our goal is to foster an open, collaborative community that includes contributions of all forms from everyone and encourages constructive, positive discourse about and the implementation of highly usable open source modeling and simulation tools with an license that supports wide adoption of the platform.

Many ways to collaborate:

- Use ICE
- Test ICE
- Give us some code
- Give us *something to* code
- Provide feedback
- Develop documentation
- Tell your friends about us
- Introduce us to some new technologies

Also, we have a small amount of funding available to support good ideas or interesting work.

Getting Started as a User

1. Join the user mailing list! - https://dev.eclipse.org/mailman/listinfo/ice-users
2. Download ICE - http://sourceforge.net/projects/niceproject/files/nightly/nice/
3. Check out our wiki - http://sourceforge.net/p/niceproject/docs/Main%20Page/
4. Email the list if you have questions!

Getting Started with Development

1. Join the dev mailing list! - https://dev.eclipse.org/mailman/listinfo/ice-dev
2. Download Eclipse - http://www.eclipse.org/downloads/
3. Setup your environment - http://wiki.eclipse.org/ICE_Build_Instructions
4. Start playing with org.eclipse.ice.tablecomponenttester
5. Email the list if you have questions!

Any Questions?

Catch the YouTube Videos! Thanks to our sponsors!

Sourceforge.net or Eclipse.org

niceproject.sourceforge.net
www.eclipse.org/ice

YouTube

youtube.com/jayjaybillings

Ohloh.net

ohloh.net/p/niceproject

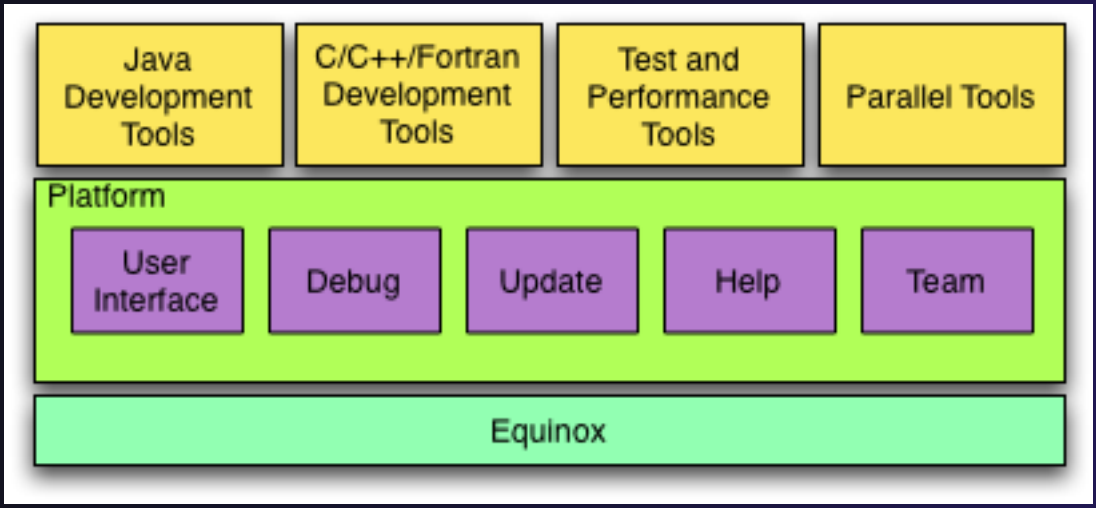
Additional Authors: Andrew Bennett, Jordan Deyton, Dasha Gorin, S. Forest Hull, Alexander J. McCaskey, Taylor Patterson, Claire Saunders, Matthew Wang, Anna Wojtowicz

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

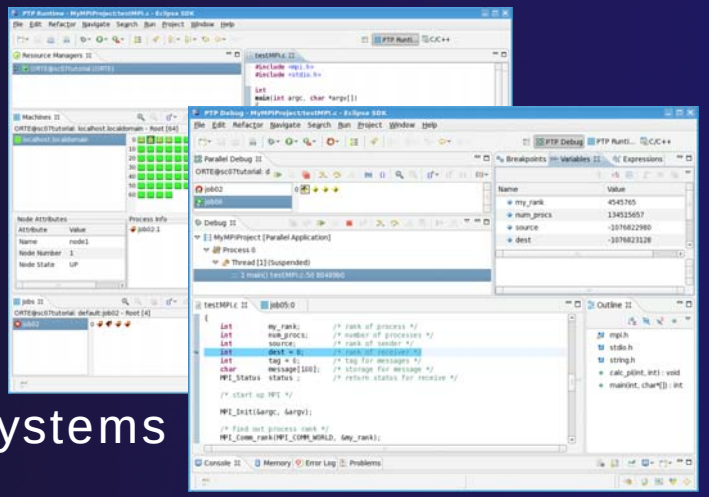


Eclipse Features

- 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

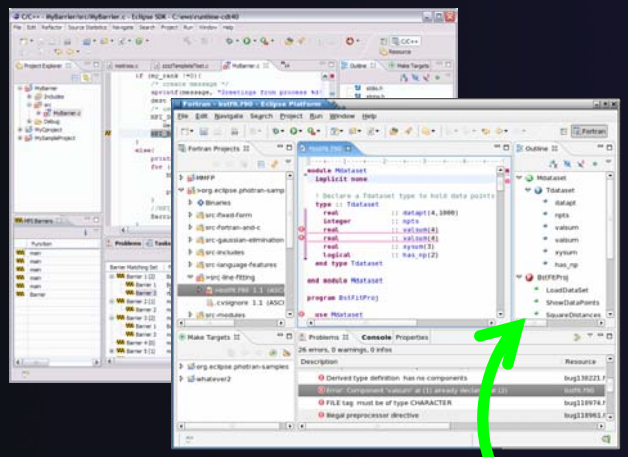
Parallel Tools Platform (PTP)

- 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

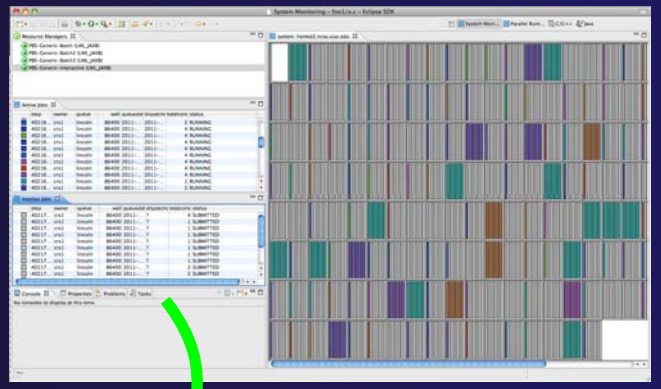


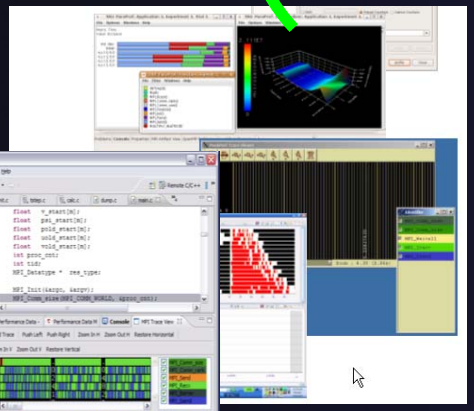
Eclipse PTP Family of Tools

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

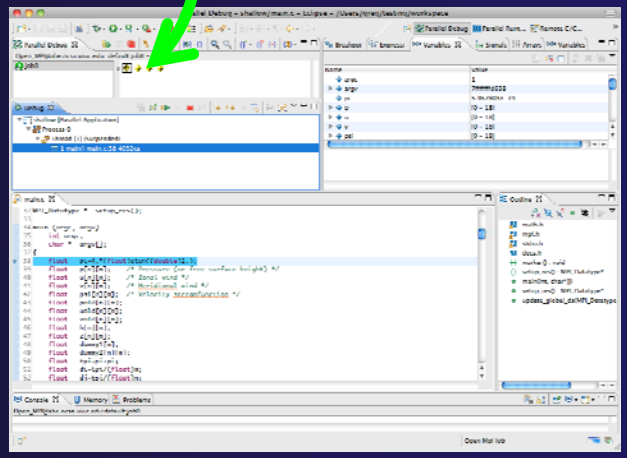


Launching & Monitoring



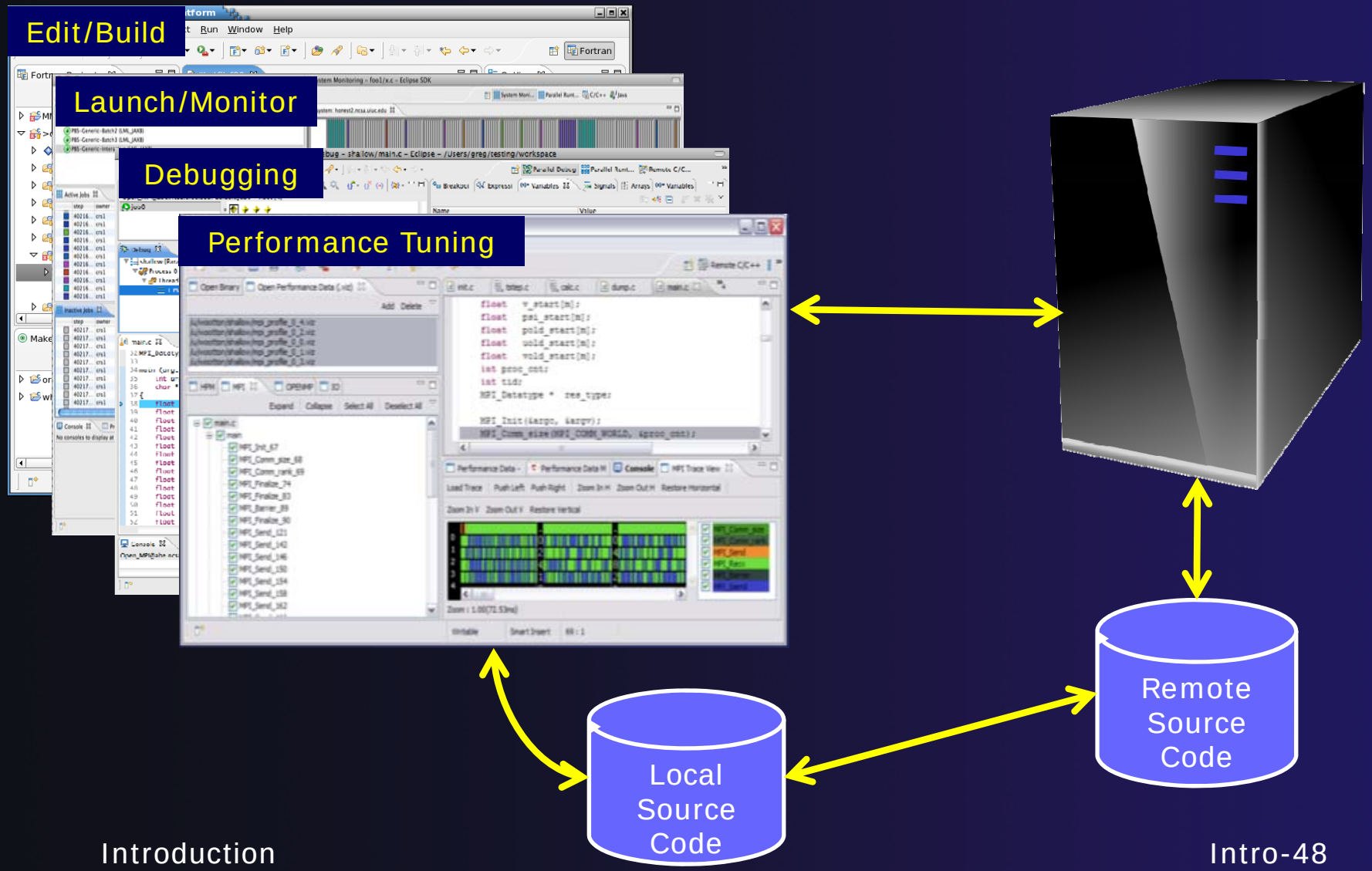


Performance Tuning
(TAU, PerfSuite, ...)



Parallel Debugging

How Eclipse is Used



Final Slides, Installation Instructions for Eclipse PTP

- Please go to http://wiki.eclipse.org/PTP/tutorials/XSEDE15 for full set of slides and installation instructions from half day tutorial at XSEDE15

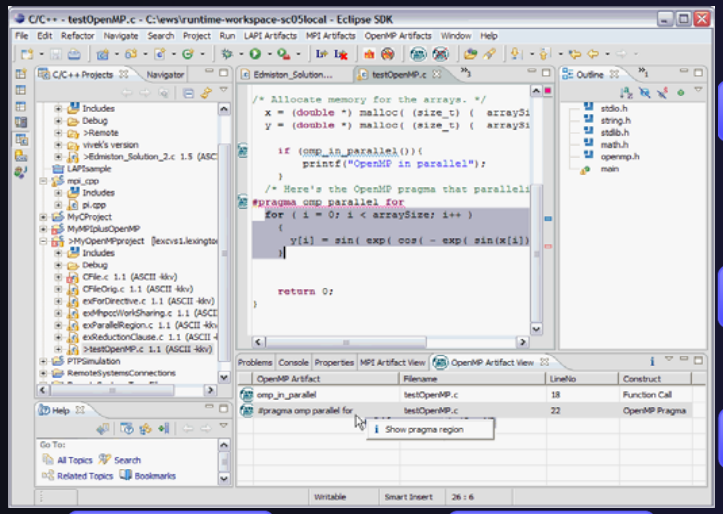
Creating a Synchronized Project

- Objective
  - Learn how to create and use synchronized projects
  - Learn how to create a sync project
    - From a source code repository in Git
- Contents
  - Eclipse project types
  - Clone a git repository; create a synchronized project
  - Using synchronize filters
  - Remote Terminal view

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

(Source Code Repositories)

- Eclipse supports a range of revision control systems, such as CVS, Git, and Subversion (and others)
- These are distinct from synchronized projects
- Revision control systems 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 source code repository (Git) <- this tutorial
- Remote -> Local
  - Projects start out on remote machine then are synchronized to the local system
  - Two options
    - Already on remote system
    - Checked out from source code repository

C, C++, and Fortran Projects

Build types

- Makefile-based
  - Project contains its own build command – typically a makefile (or makefiles) for building the application – but can be any build scripts, etc.
- Managed
  - Eclipse manages the build process, no makefile required by the user

Check out source code from Git repository

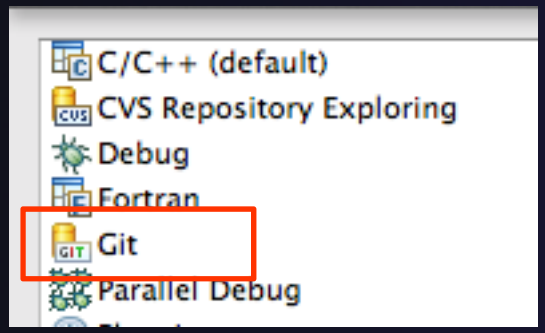
Create Synchronized project on the local machine at the same time.

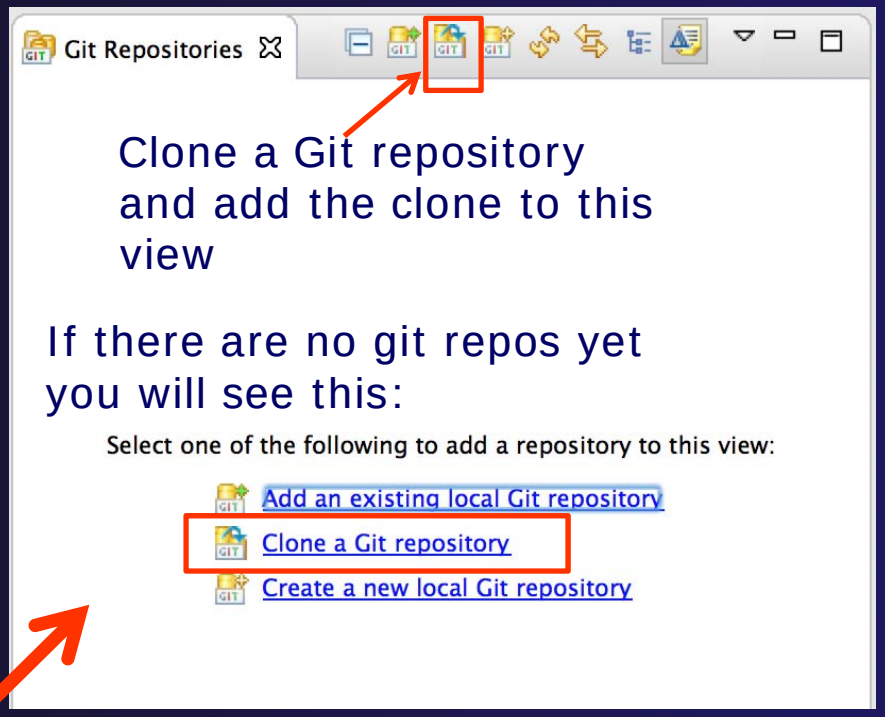
Two steps:

- Clone Git Repo
- Create project files from within the clone

Clone the git repo

- Open Git perspective
  - Window > Perspective > Open Perspective > Other
- Select Git



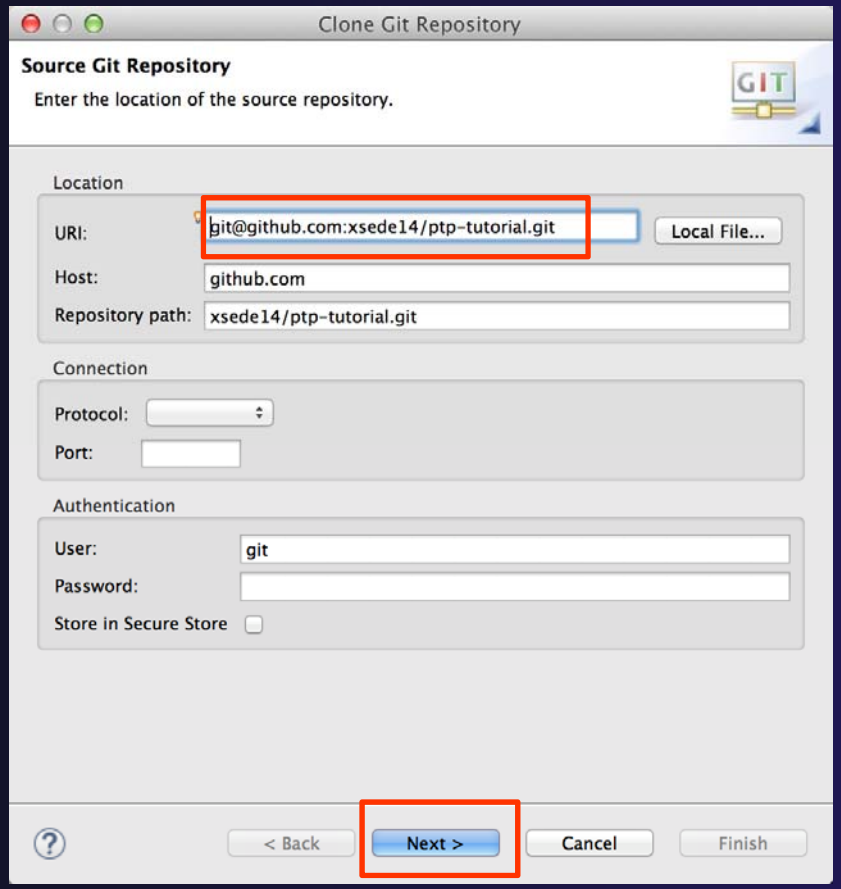


- In the view, select **Clone a Git repository** one of two ways

Specify remote git repo location

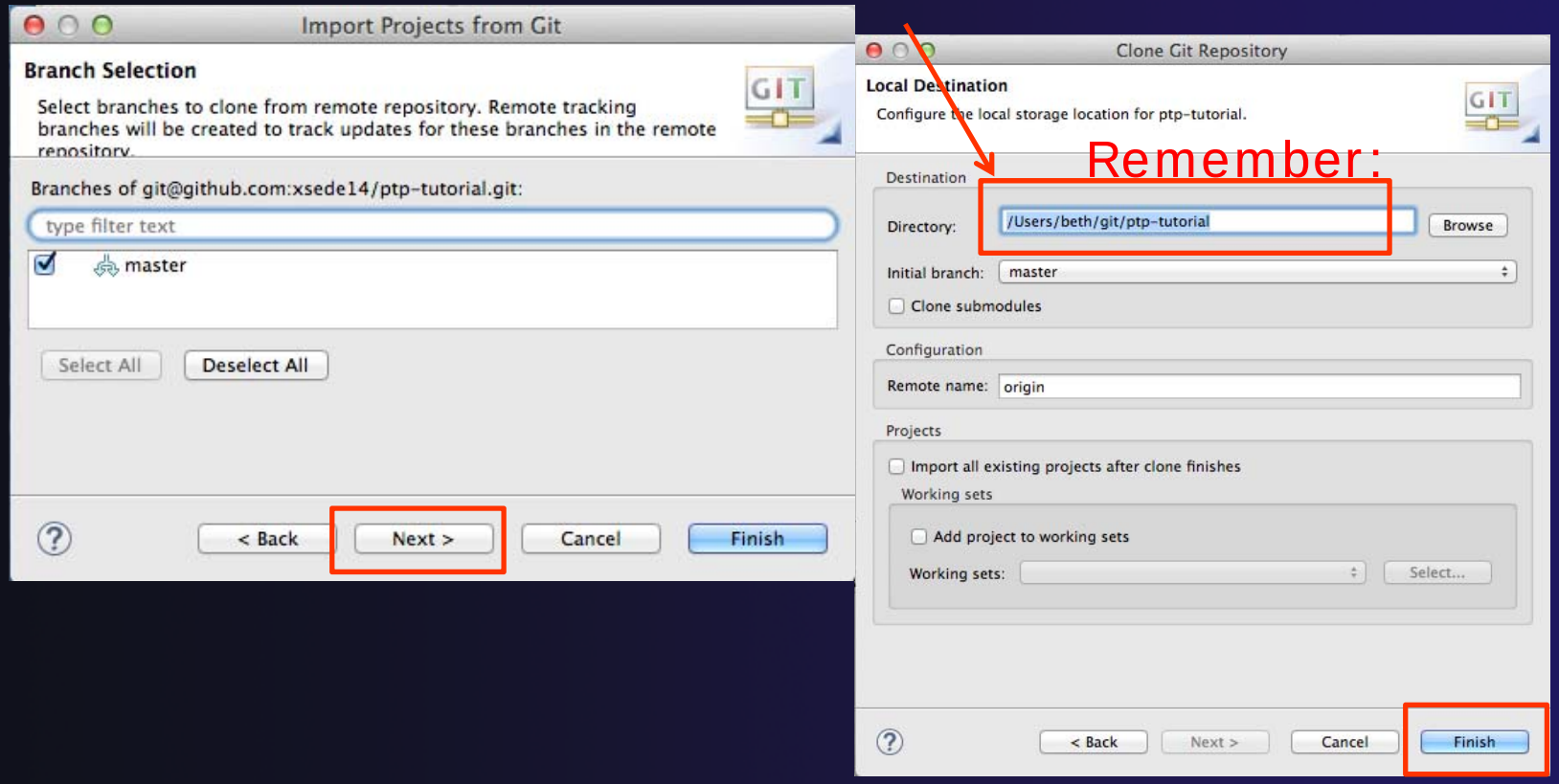
- URI: https://github.com/xsede14/ptp-tutorial.git

- Fill in URI and other fields fill themselves
- Select **Next>**



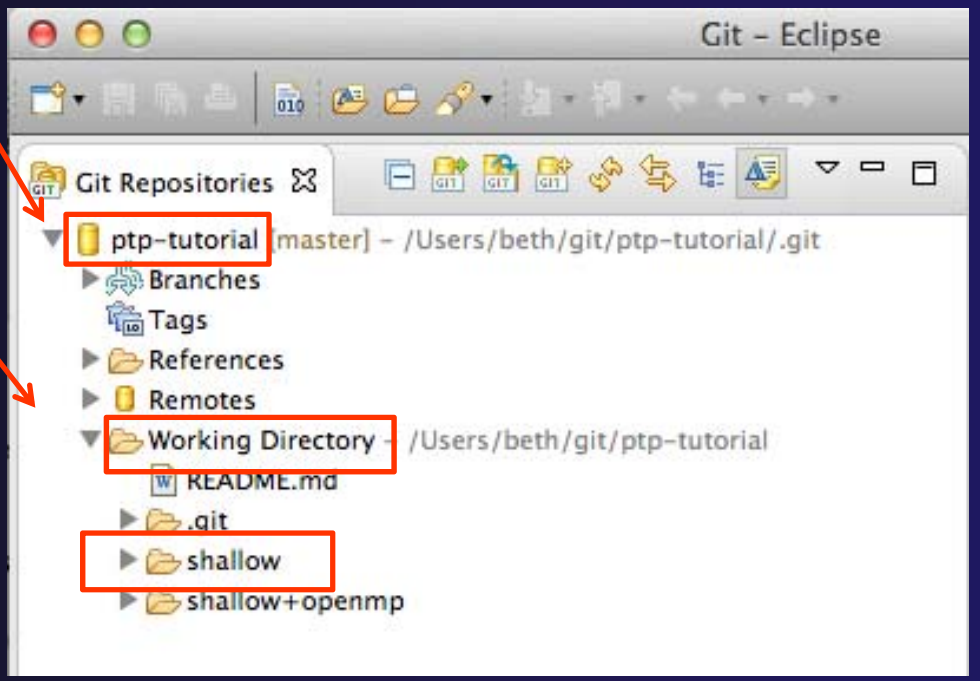
Finish git cloning

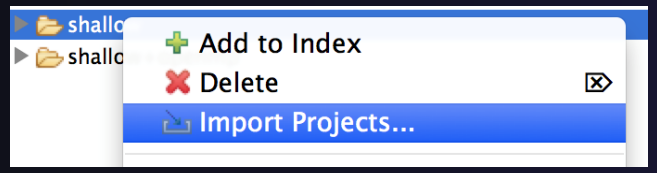
- Select **Next>** to choose the (only) branch
- Then select **Finish>** to use the default git destination (Remember this, you'll need it later)



Import project from cloned repo

- After repo is cloned, expand ptp-tutorial and Working Directory
- We are importing only one project
- Select **shallow**
- Right mouse, **Import Projects...**

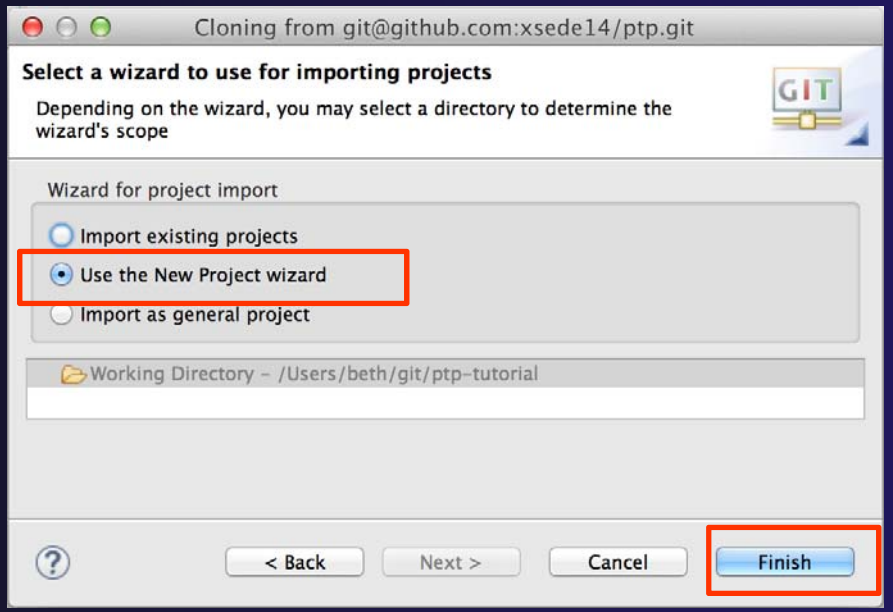




Create new project with wizard

- Select **Use the New Project Wizard** to be able to create the project as a Synchronized C/C++ project at creation

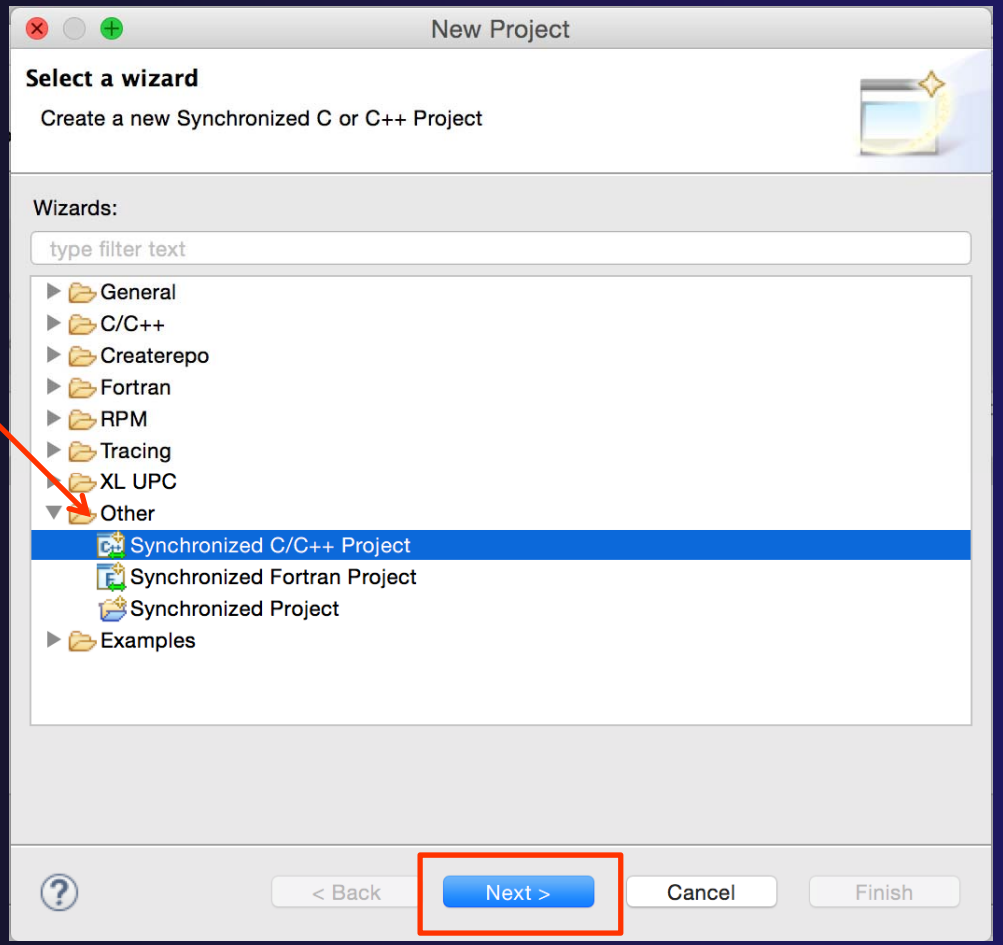
- Select **Finish** to finish the git cloning, and you will be taken to Sync project info next.



New Project Wizard

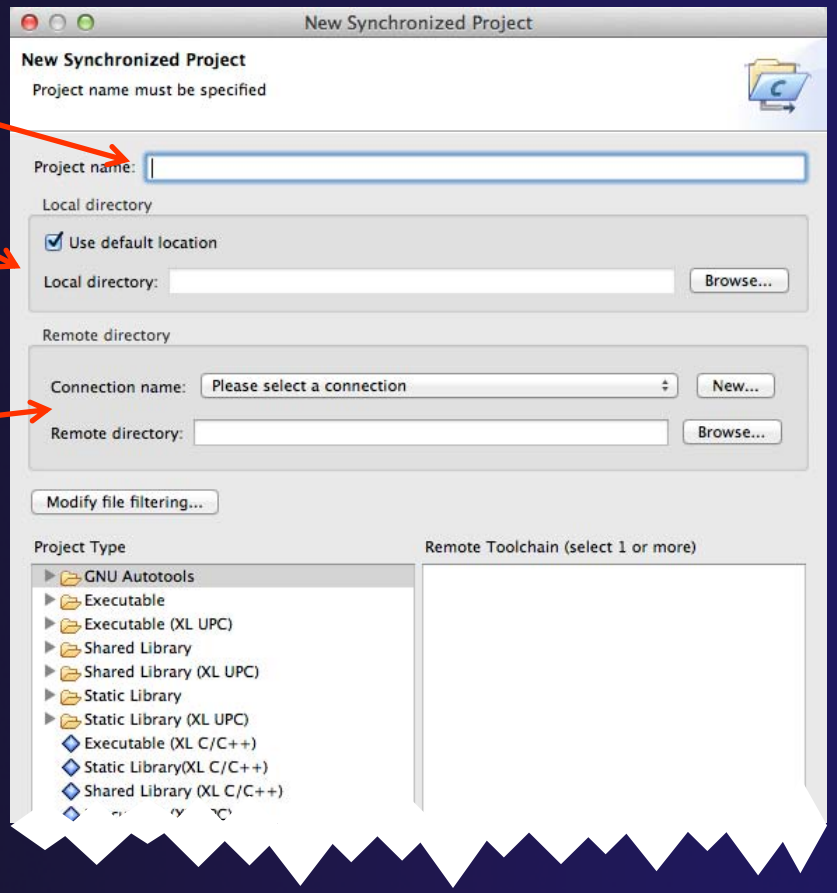
We are creating the project directly as a Synchronized C/C++ project

- Expand **Other**
- Select **Synchronized C/C++ Project**
- Select **Next>**



New Synchronized Project Wizard

- Enter the **Project Name**
  - E.g. “shallow”
- Next we will specify the **Local Directory** where the local files are located (cloned from git)
  - Files are synchronized here, and we will edit them locally
- ...and the **Remote Directory** where the remote files are located
  - Our remote target machine, where we will build, run, & debug
- Use **Modify File Filtering...** if required (see later slide)

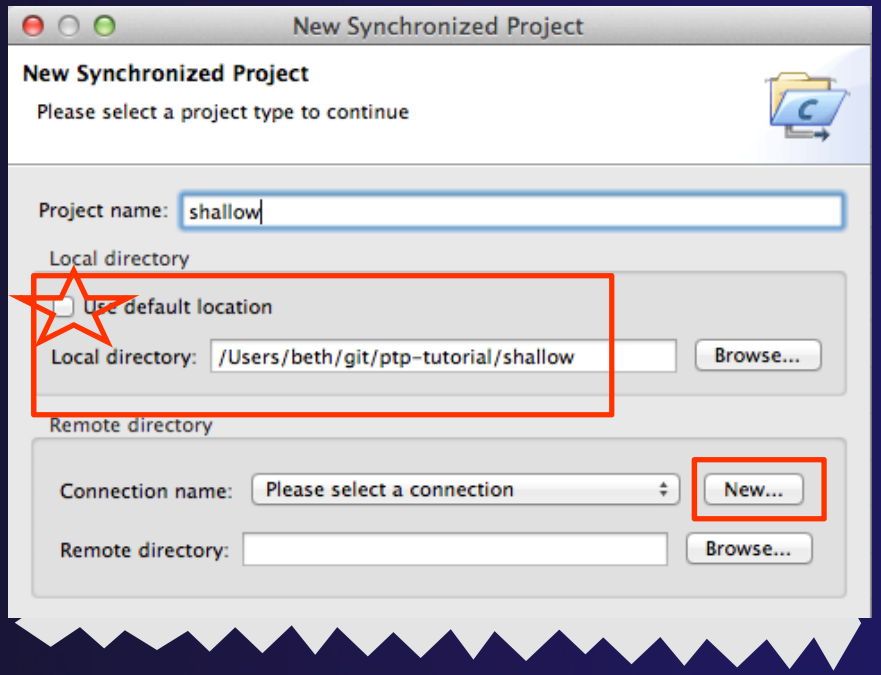


See Next slides...

Local and remote directories

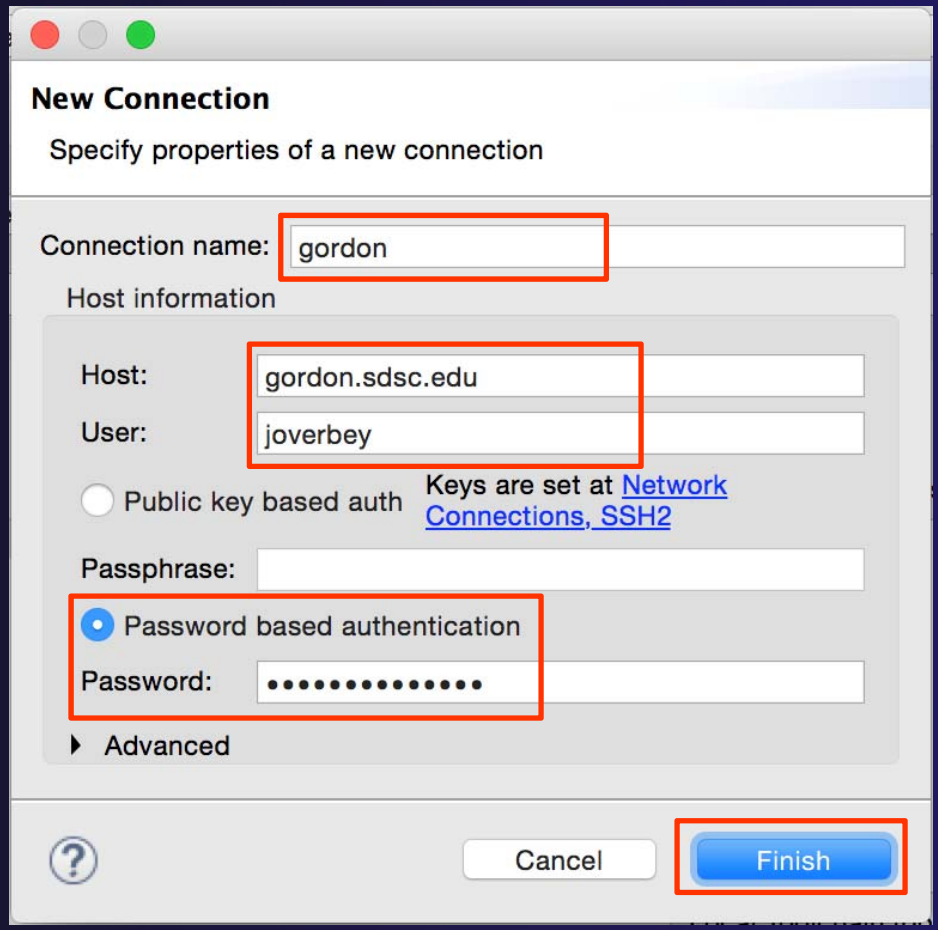
1. For Local directory,
   NOTE: Uncheck **Use default location**
   and browse to the location you chose for git repo
   - the **shallow** dir beneath that

2. To specify the Remote directory, first Create a connection to the remote target machine by selecting **New...**



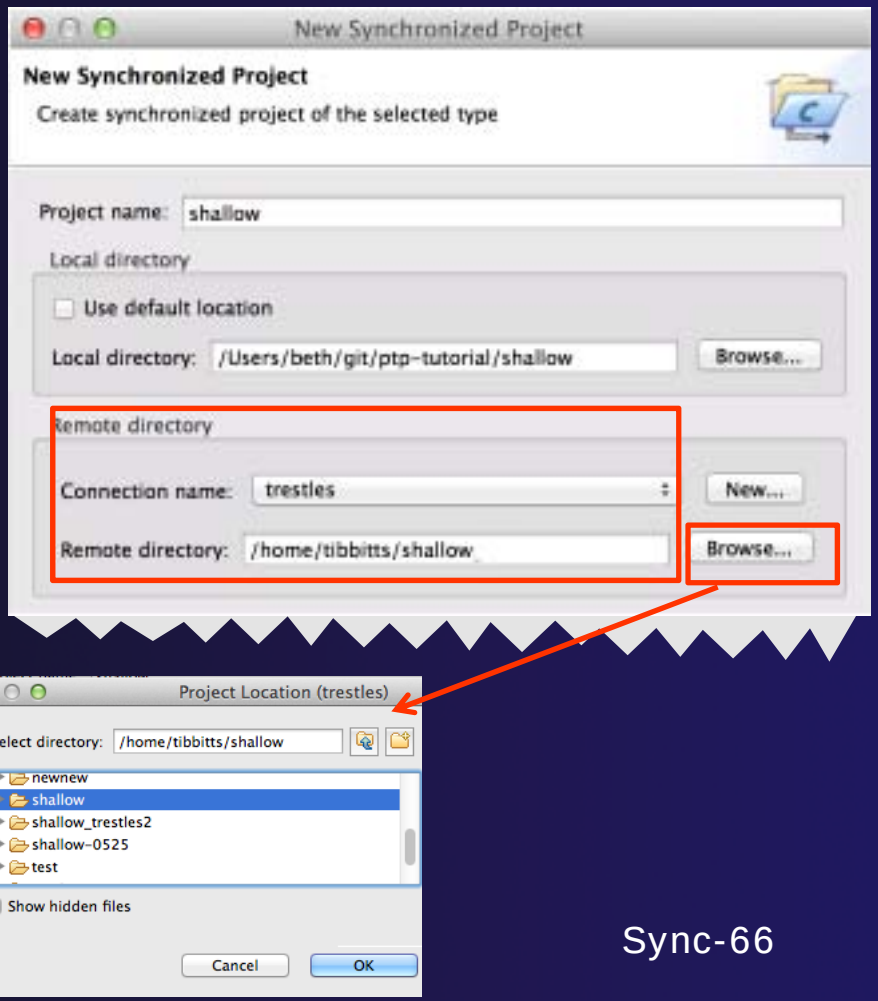
Creating a Connection

- In the **New Connection** dialog
  - Enter a **Connection name** for the remote host
  - Enter host name, user name, and user password or other credentials
  - Select **Finish**



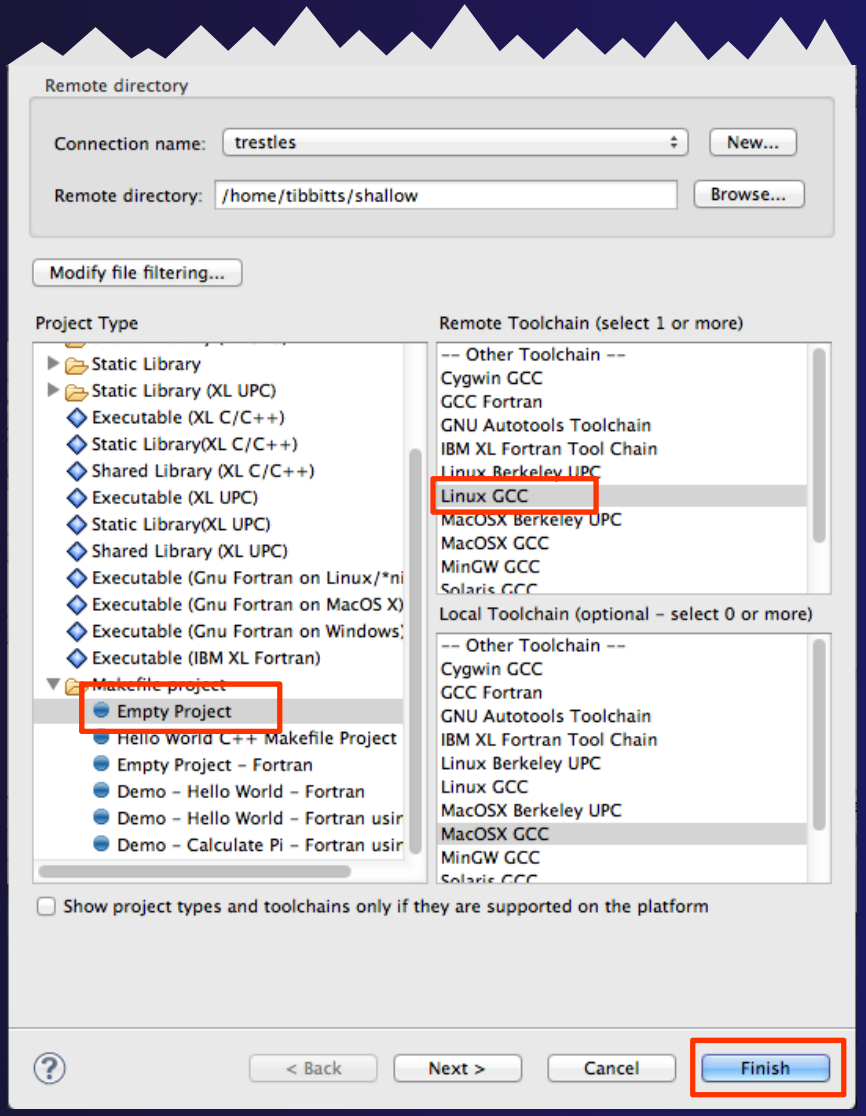
Specifying the remote directory

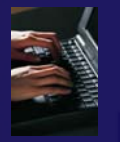
- After the connection has been specified, back in the **New Synchronized Project** window..
- For Remote directory, you can enter its location. If it does not exist, it will be created.
  - If the remote dir exists, you can select it with the **Browse...** Note that this is the first time that the Connection information is utilized.
- *Later slides in this section show how to fix Connection if e.g. password or userid are entered incorrectly*



Project Type & Toolchain

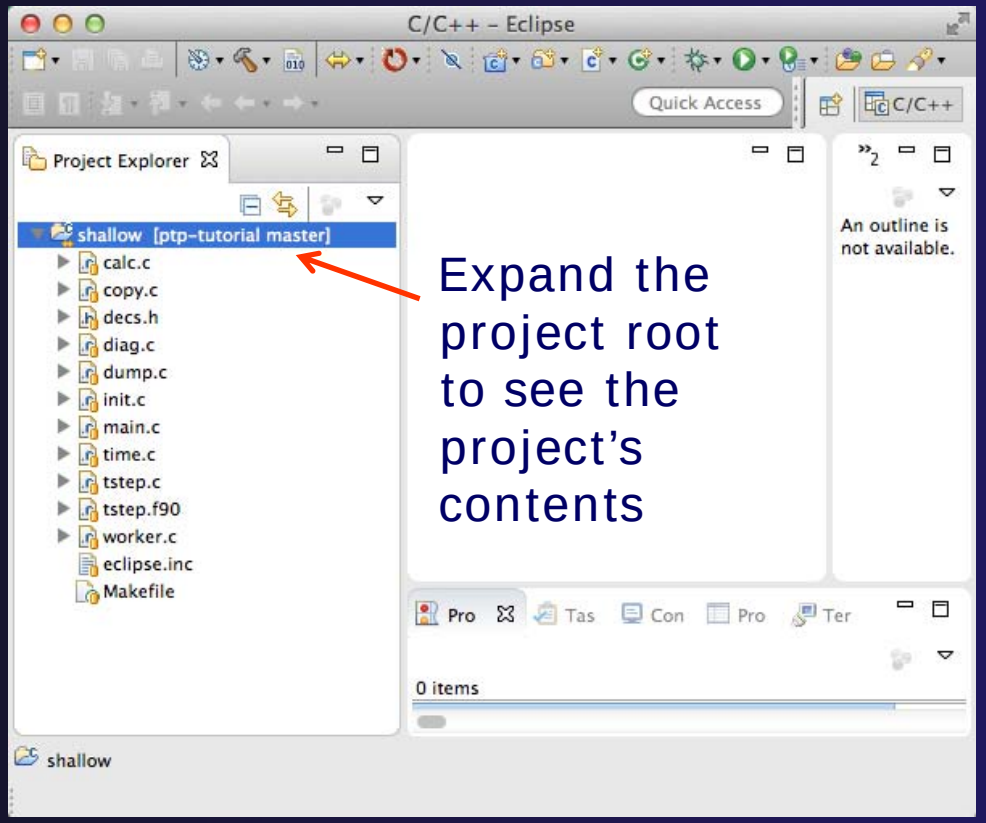
- Choose the **Project Type**
  - This tutorial's code has its own makefile, so use **Makefile Project>Empty Project**
  - Otherwise, choose the type of project you want to create
- Choose toolchain for remote build
  - Use a toolchain that most closely matches the remote system
- Choose a toolchain for the local build (OPTIONAL)
  - This is optional if you don't plan to build on the local machine
  - This is used for advanced editing/searching
- Click **Finish** to create the project



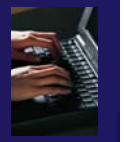


Project successfully created

- You should now see the “shallow” project in your workspace
- Project is synchronized with remote host

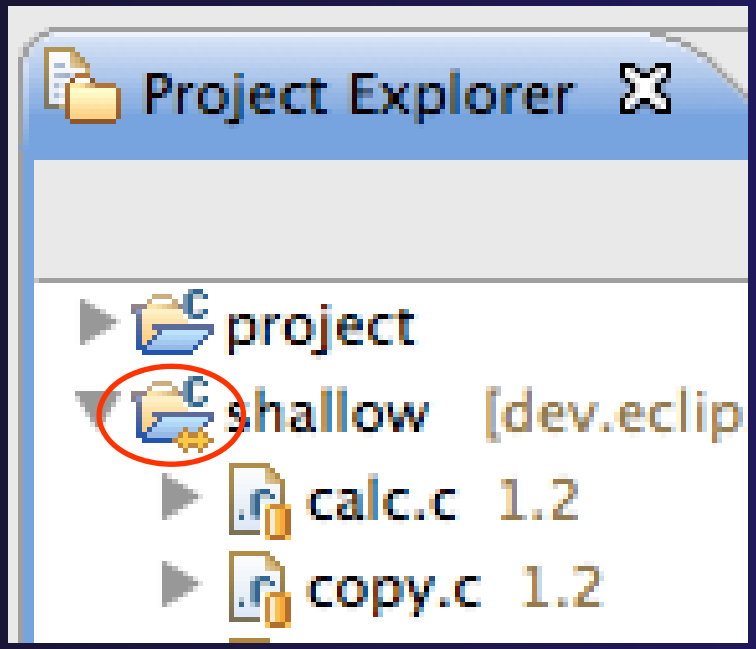


Status area in lower right shows Synchronization progress:



Synchronized Project

- Back in the Project Explorer, decorator on project icon indicates synchronized project
- Double-+ icon



- C Project w/o Sync

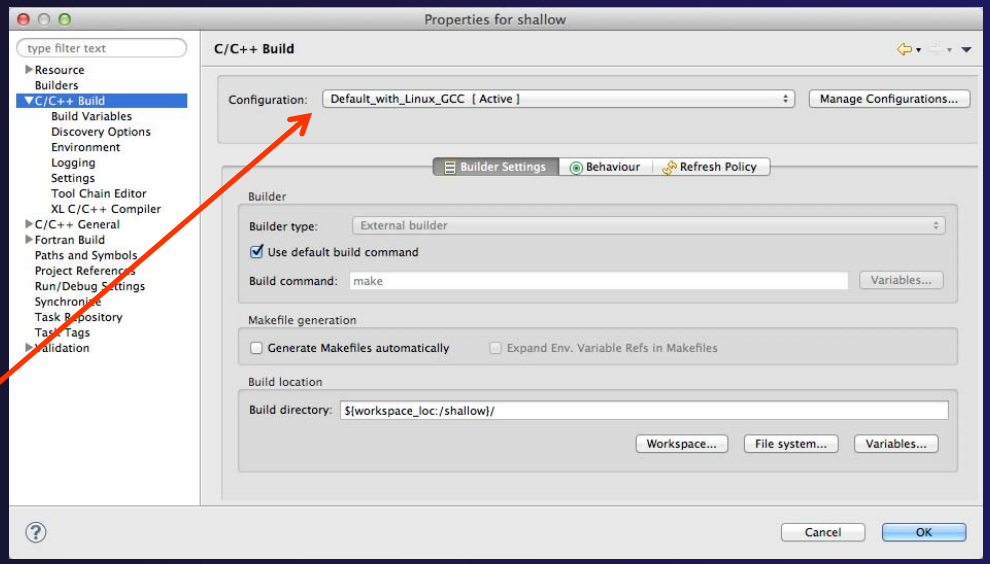
- Synchronized Project

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 do environment configuration with **modules**
  - How to create build targets

Build Configurations

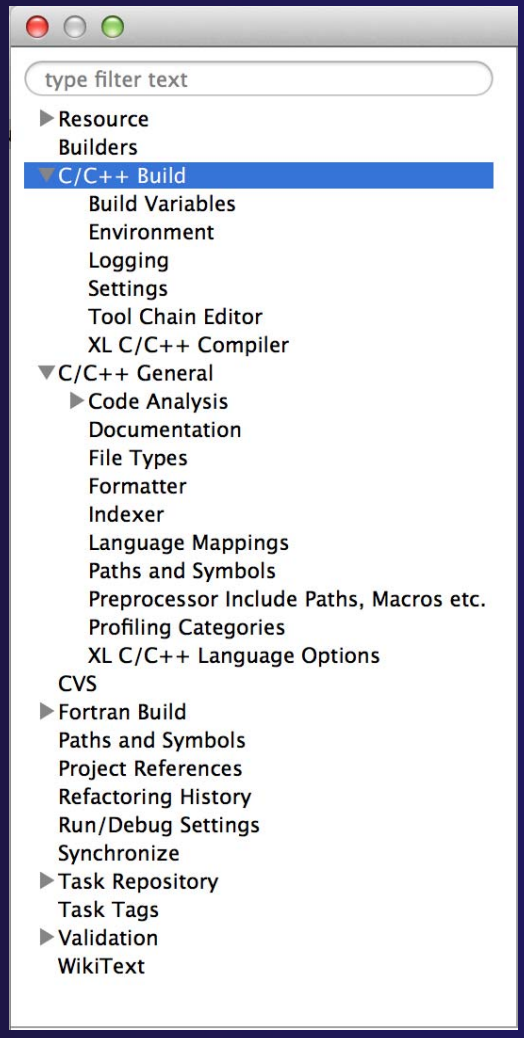
- A build configuration provides the necessary information to build the project
- The build configuration information is specified in the project properties
- Projects can have multiple build configurations, each configuration specifies a different set of options for a build
- Open the properties by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

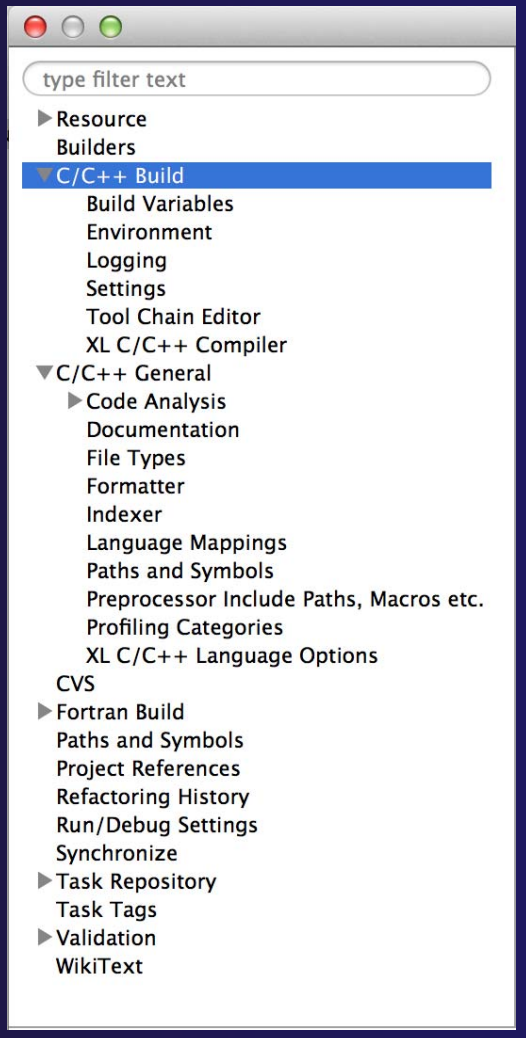
Build Properties (1)

- **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
- **Environment**
  - Modify/add environment variables passed to build
- **Logging**
  - Enable/disable build logging



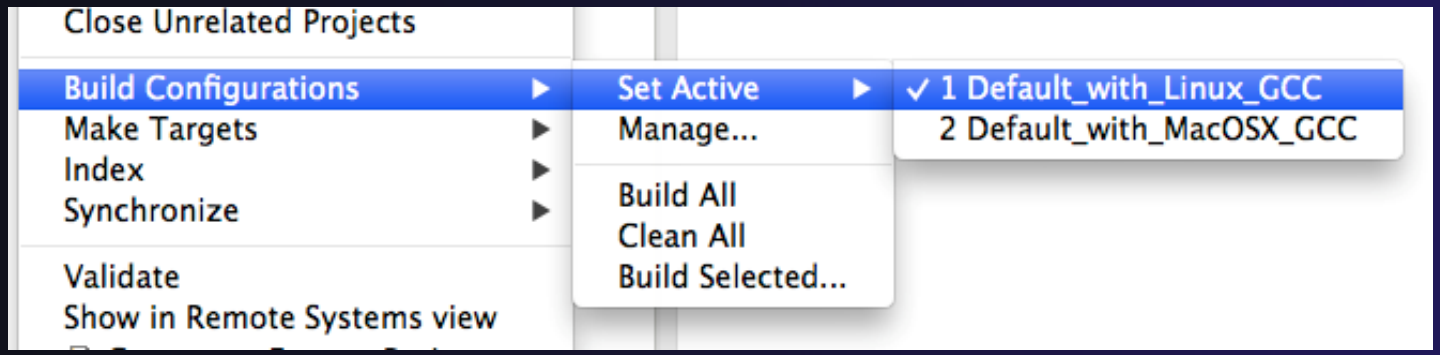
Build Properties (2)

- **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)
- **C/C++ General/Preprocessor Include Paths...**
  - Set include paths here



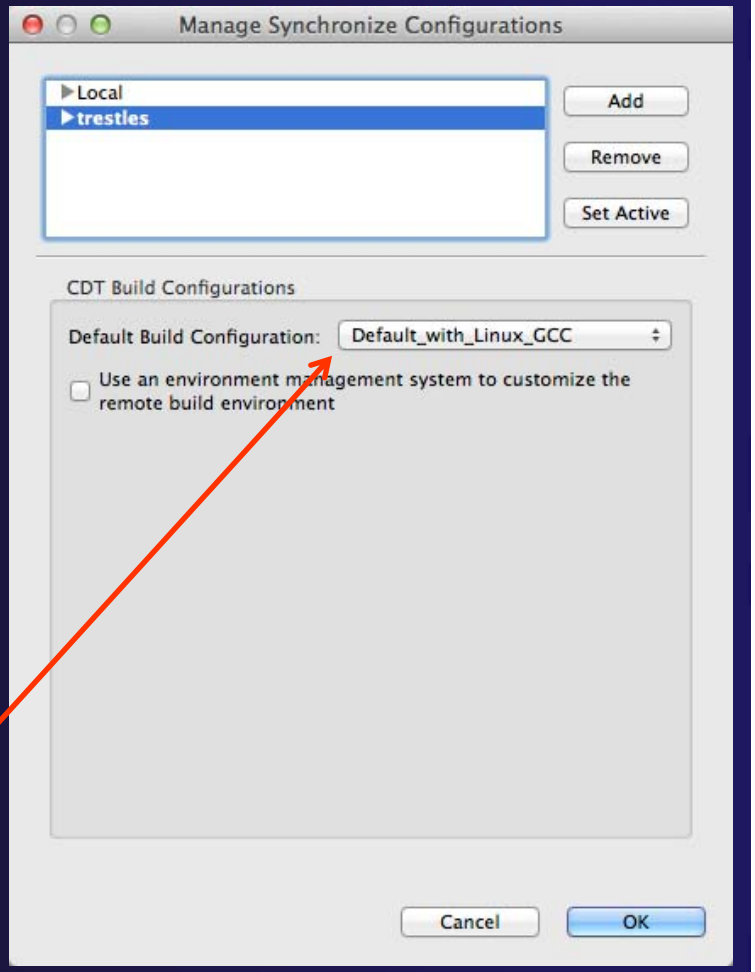
Selecting Build Configuration

- 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



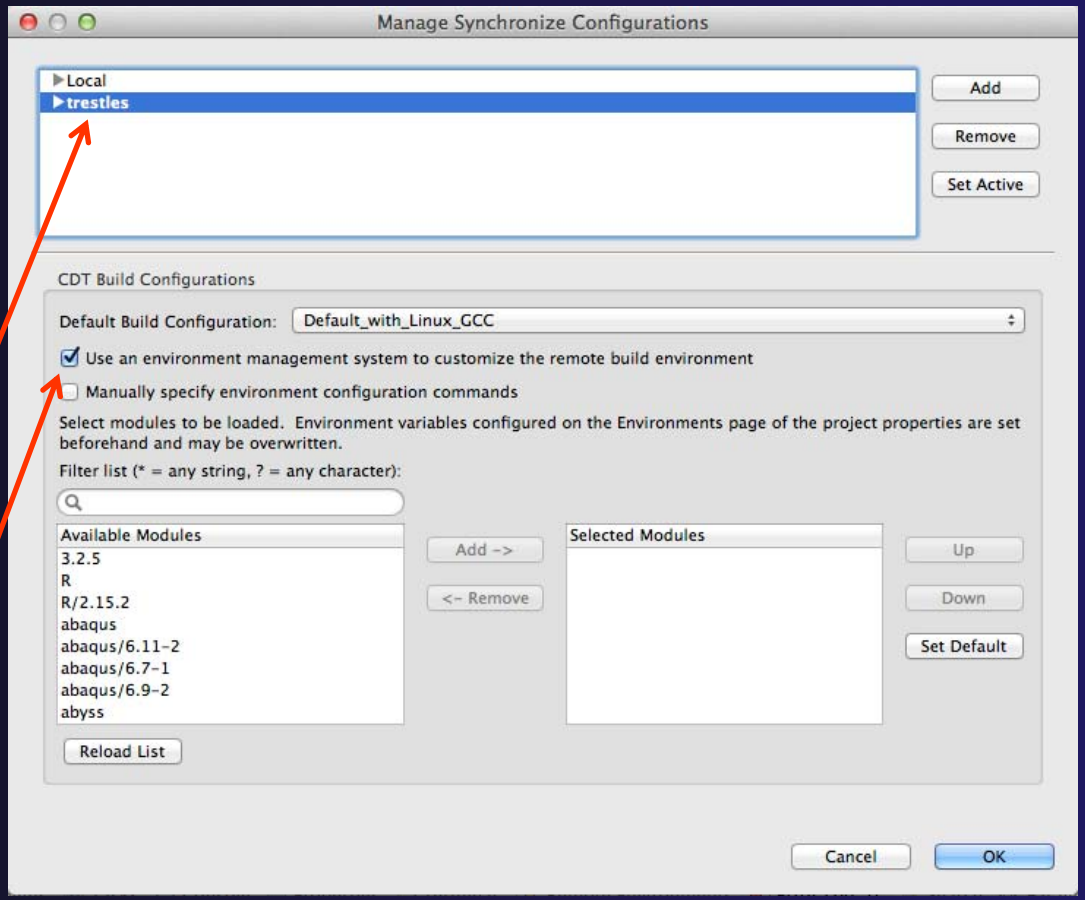
Building Synchronized Projects

- 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
- Right mouse on Project, **Synchronize > Manage…**
- A build configuration can be associated with a synchronize configuration, so that it is automatically selected when the synchronize configuration is changed



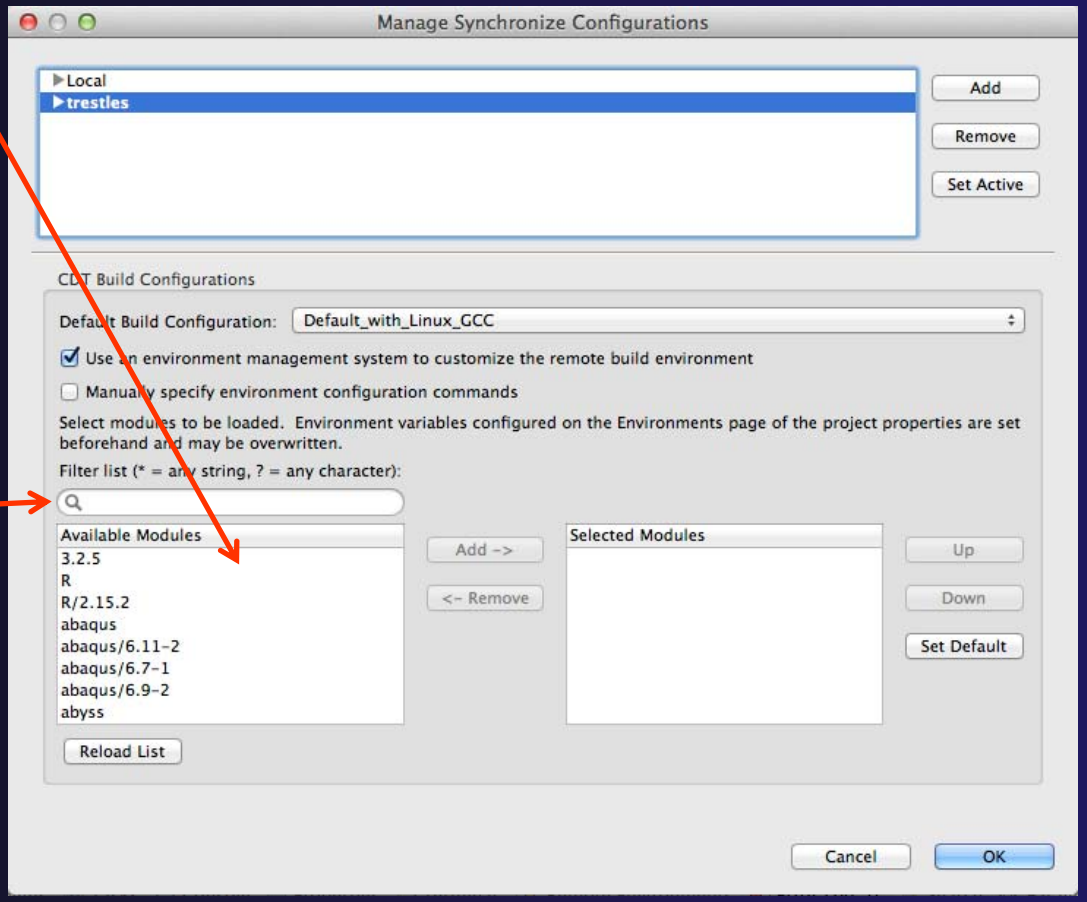
Configuring the Build Environment

- 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



We'll do this for tutorial in a few slides...

Build Environment (3)

- When you 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

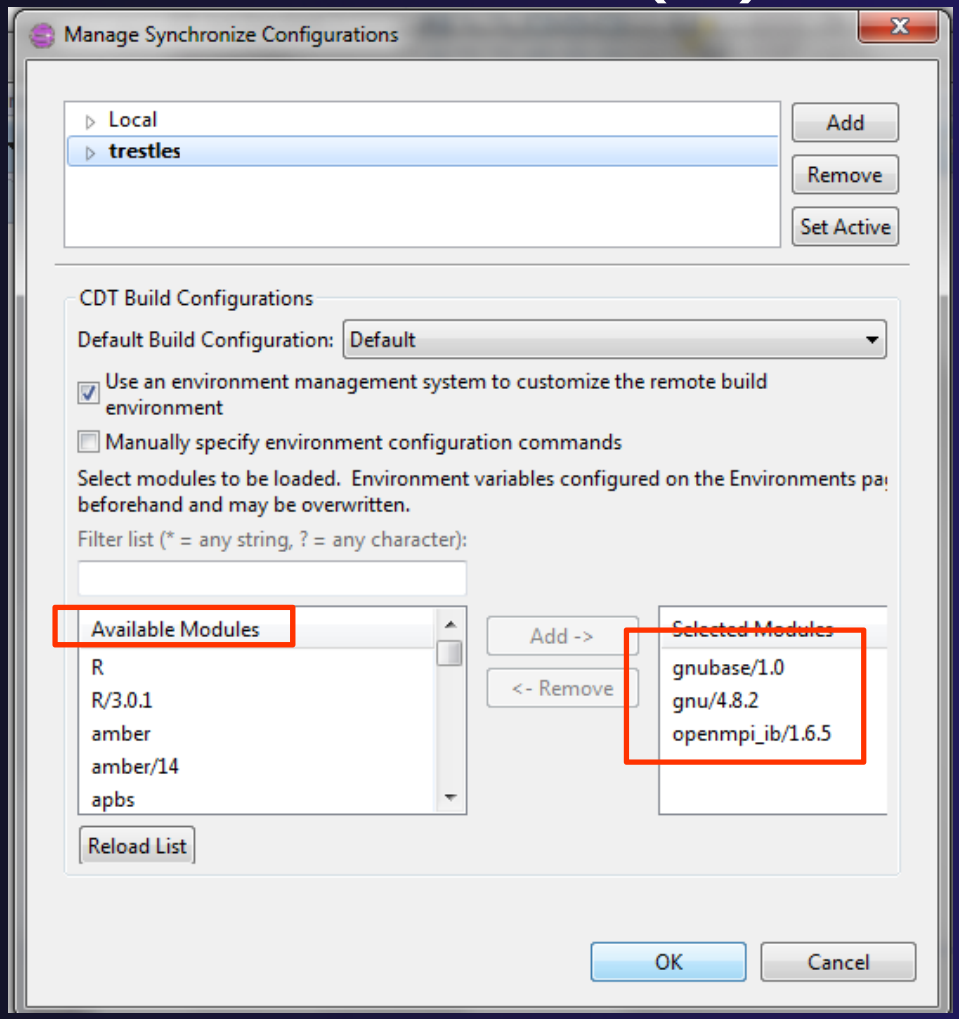
CDT Build Console [shallow]

```
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
module load globus-5.0.4-n1
```

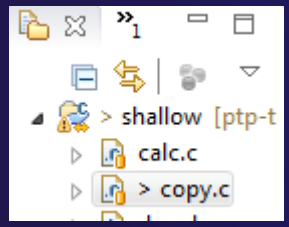
Build Environment (4)

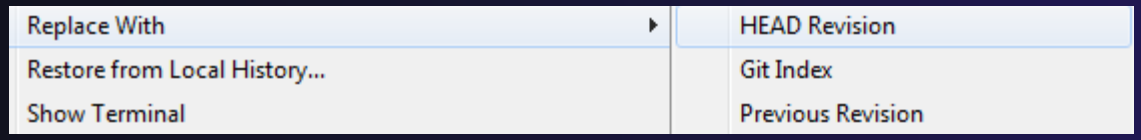
- For this tutorial, we want to use gcc and Open MPI
- To get to this dialog: Right mouse on Project, **Synchronize > Manage…**
- Navigate to **gnu** in **Available Modules** and select **Add ->**
- Navigate to openmpi_ib and select **Add ->**
- Assure the order matches this
  - If not, use **Up/Down** buttons



Start with original ‘shallow’

- Start with original ‘shallow’ code:
  - Project checked out from git:
    - Right mouse on project,
      Replace With > HEAD Revision

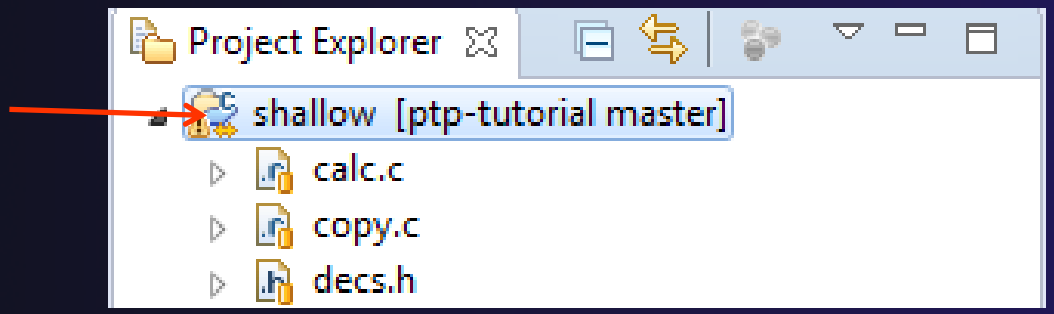




    Also see Compare With ...
  - Other project:
    - Right mouse on project,
      Restore from local history – finds deleted files
    - Right mouse on file, **Compare With**
      or **Replace With**

Starting the Build 

- Select the project in Project Explorer

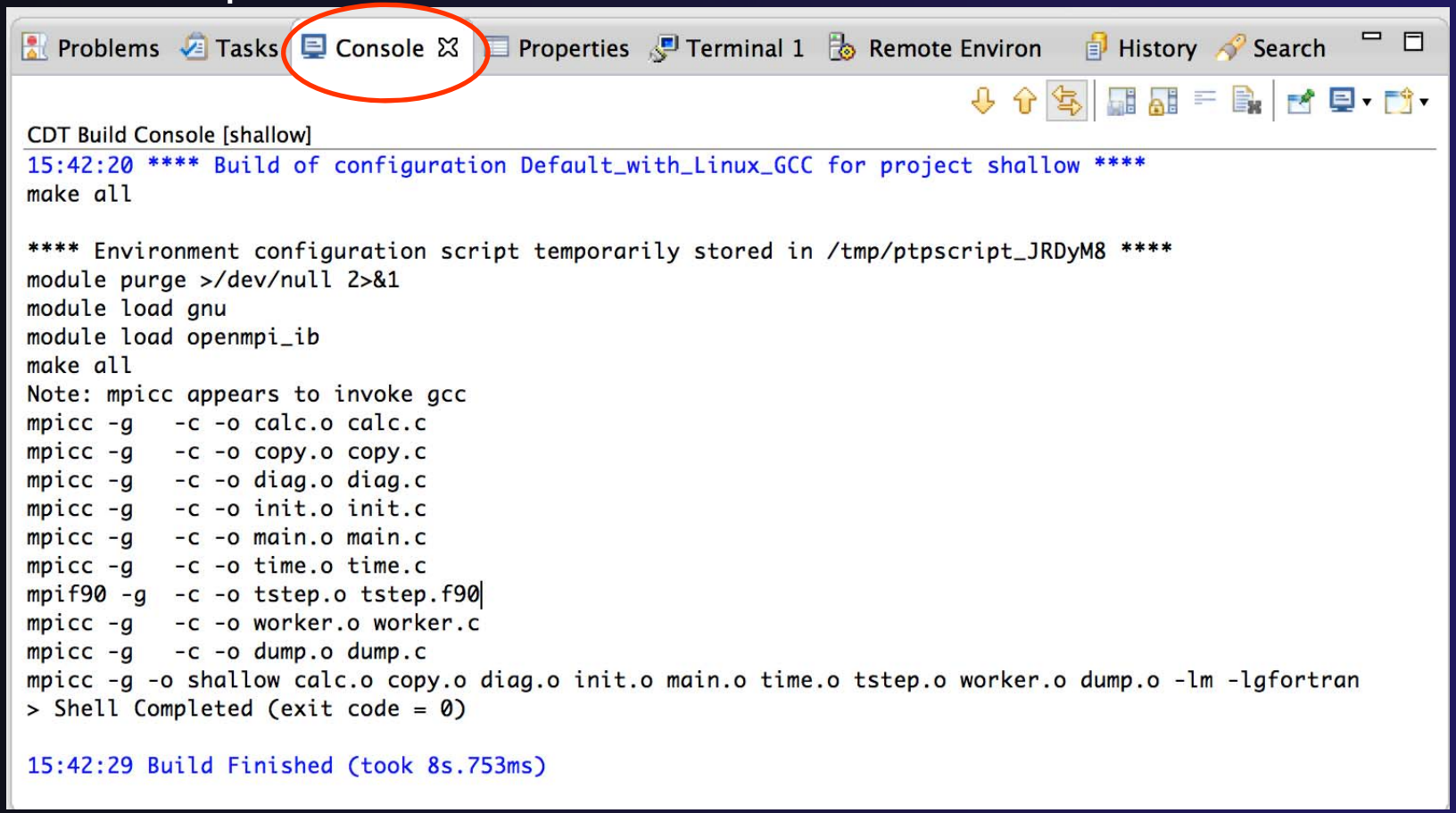


- Click on the  hammer button in toolbar to run a build using the active build configuration

- By default, the Build Configuration assumes there is a Makefile (or makefile) for the project

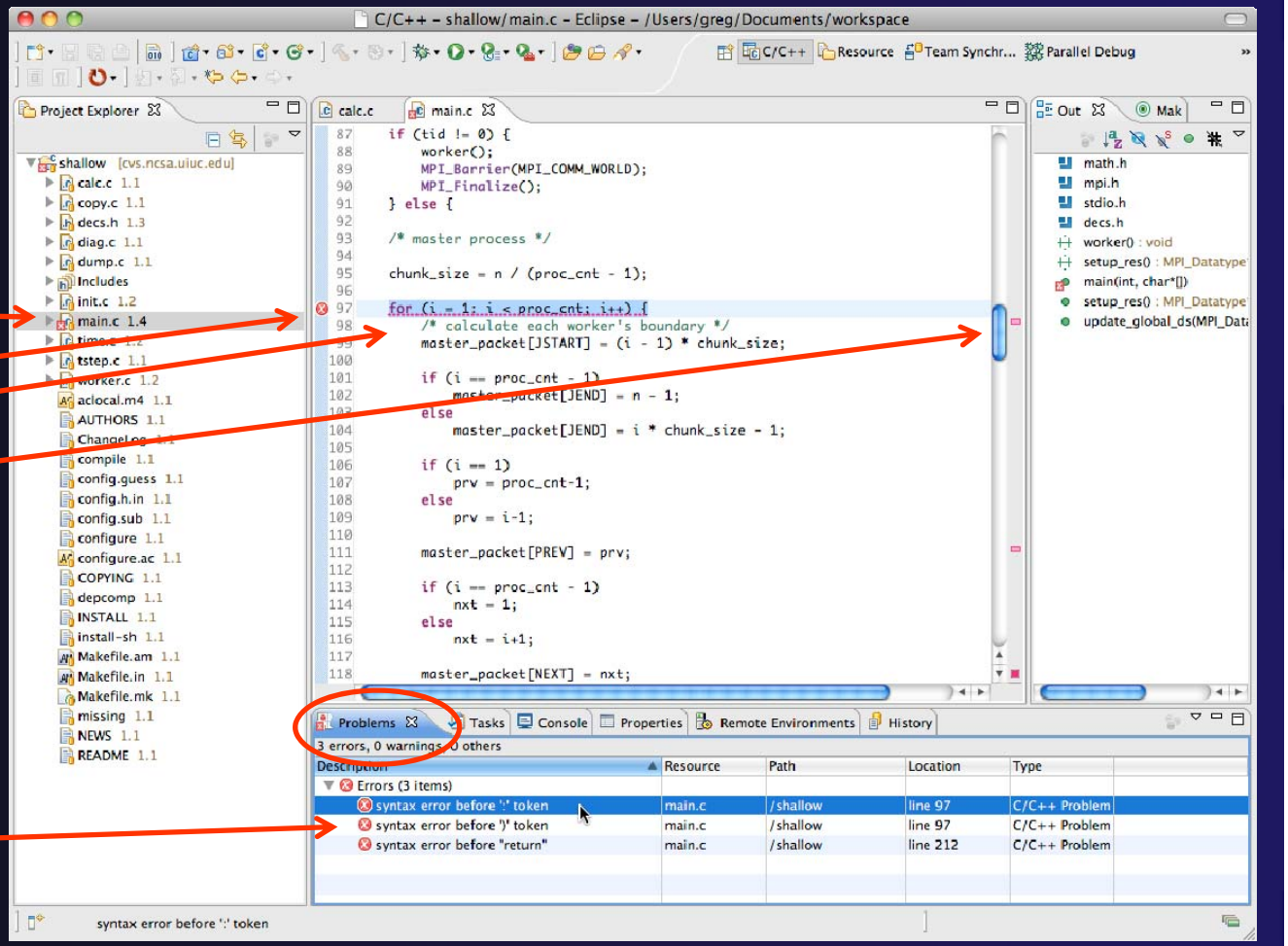
Viewing the Build Output

- Build output will be visible in console



Build Problems

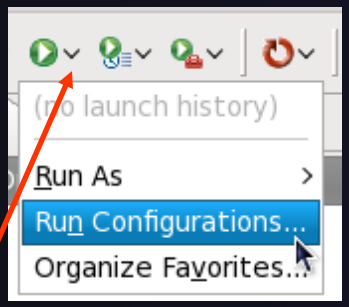
- Build problems will be shown in a variety of ways
  - Marker on file
  - Marker on editor line
  - Line is highlighted
  - Marker on overview ruler
  - Listed in the **Problems view**
- Double-click on line in **Problems view** to go to location of error in the editor



Running an Application

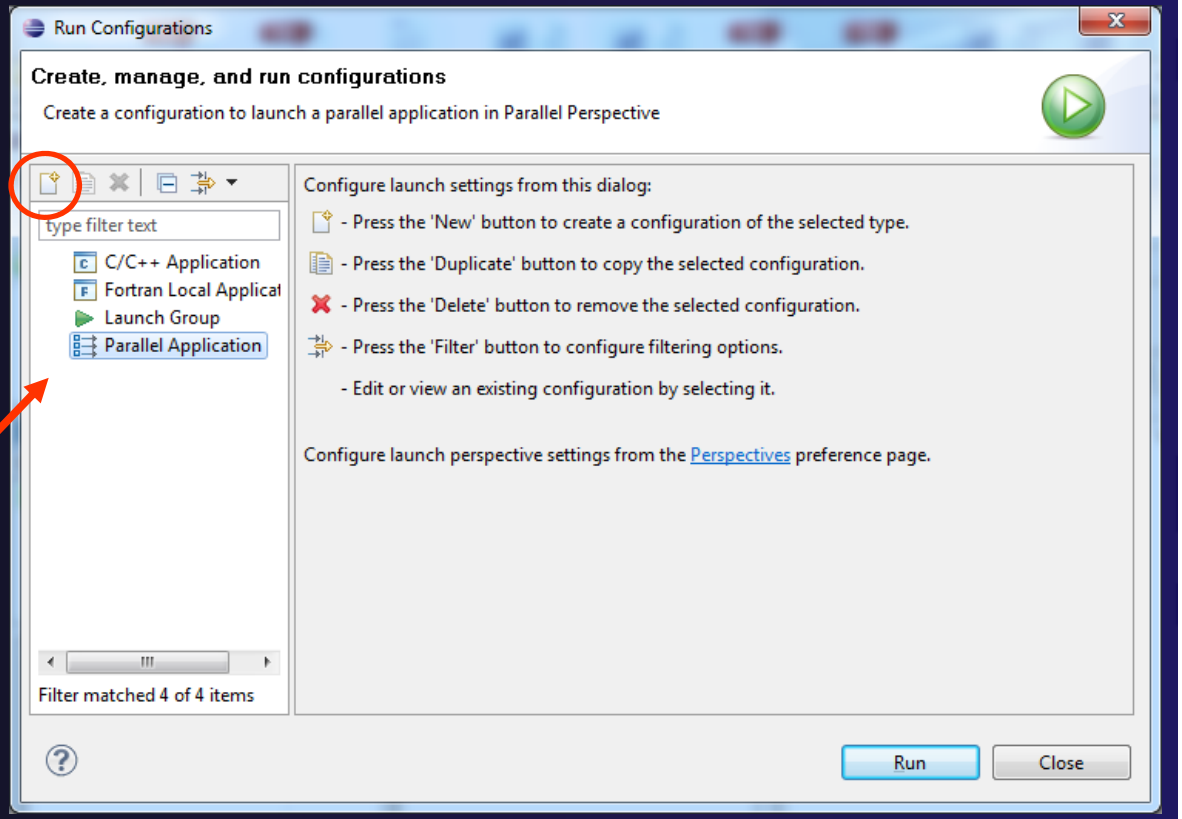
- Objective
  - Learn how to run an MPI program on a remote system
- Contents
  - Creating a run configuration
  - Configuring the application run
  - Monitoring the system and jobs
  - Controlling jobs
  - Obtaining job output

Creating a Run Configuration



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

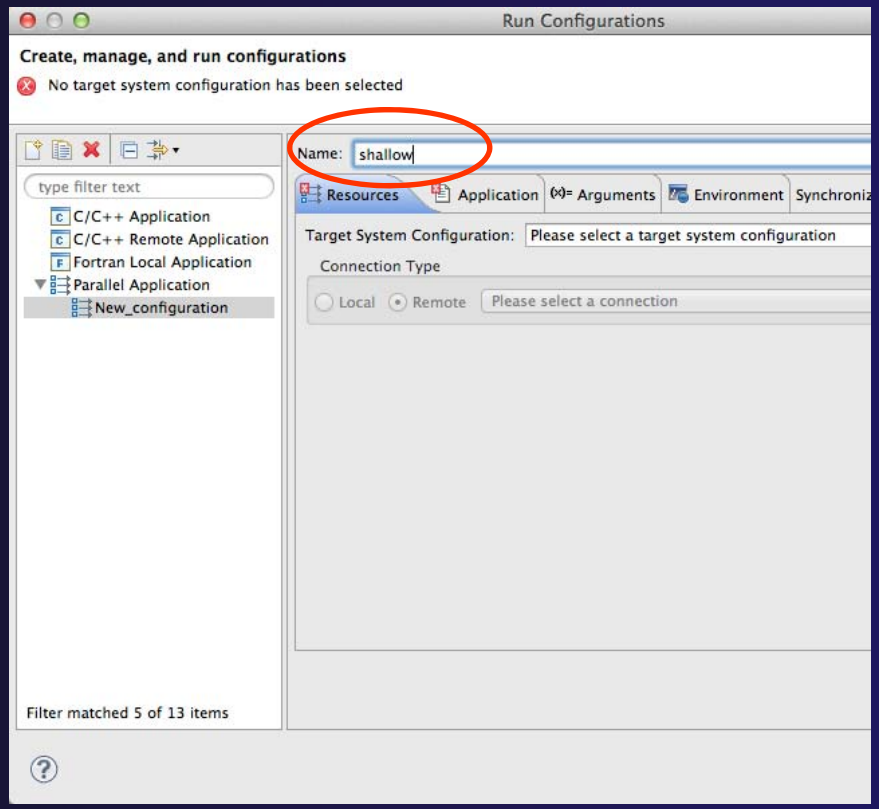
Or, just double-click on **Parallel Application** to create a new one



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.

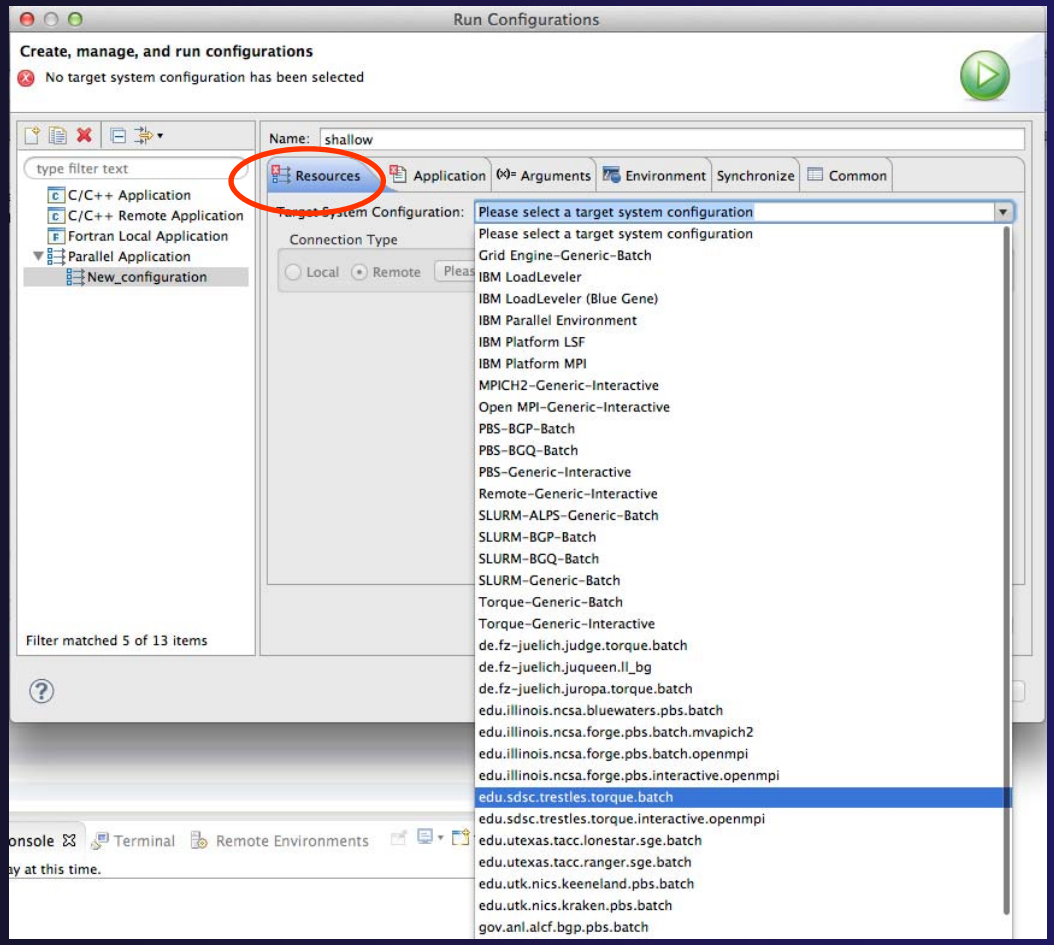
Set Run Configuration Name

- 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



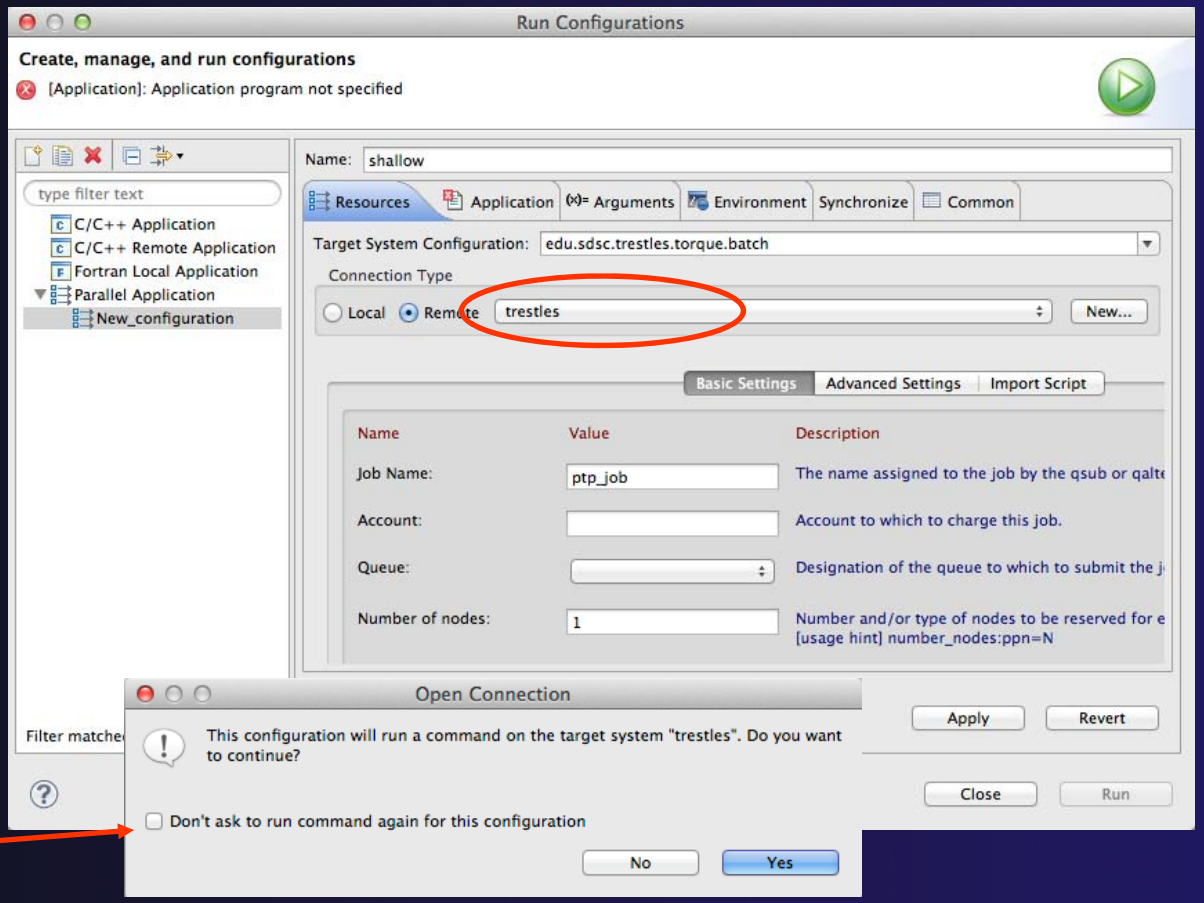
Configuring the Target System

- In **Resources** tab, select a **Target System Configuration** that corresponds to your target system
  - Use **Generic 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



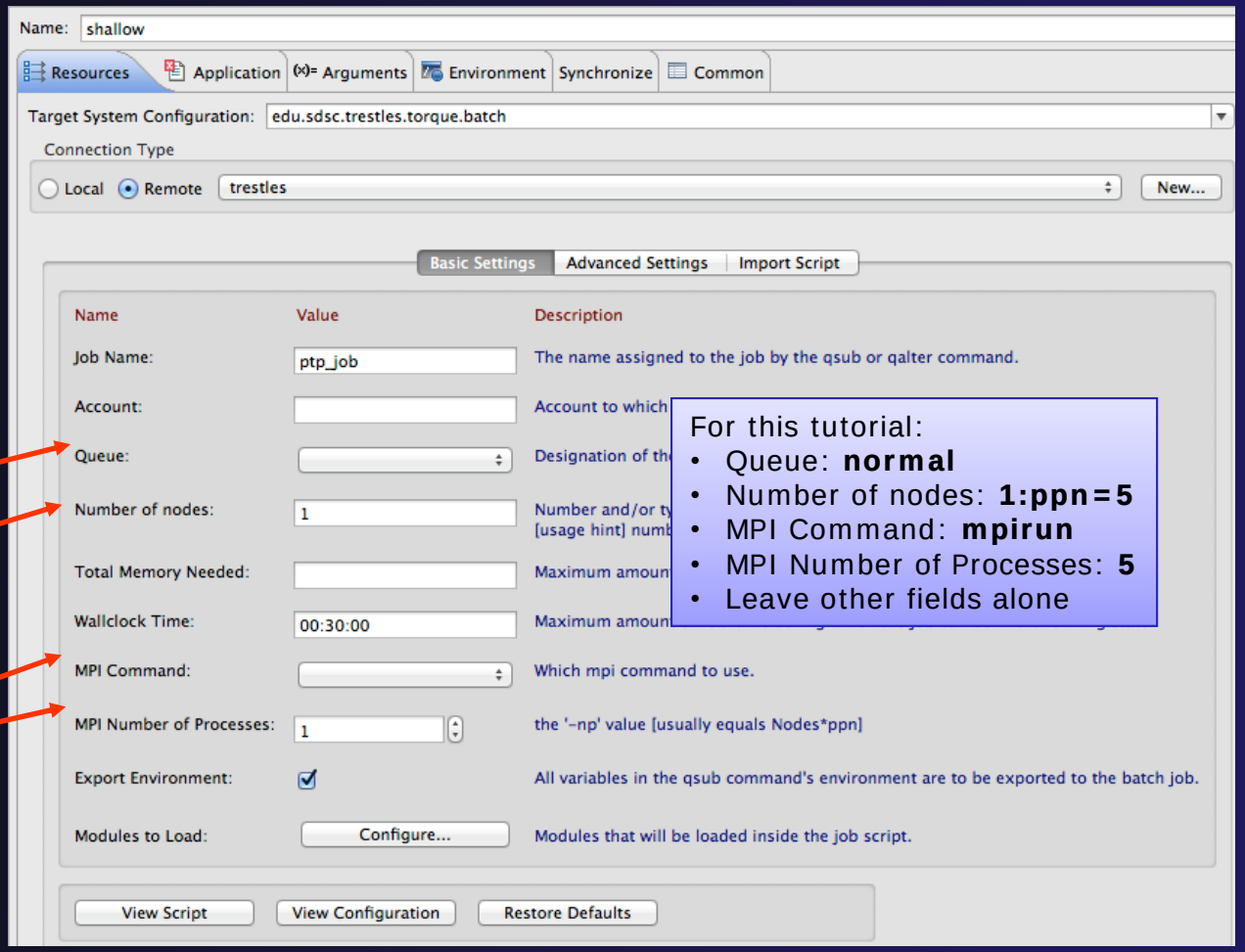
Configure the Connection

- 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 *gordon.sdsc.edu*
- Select toggle if you don't want to see popup again



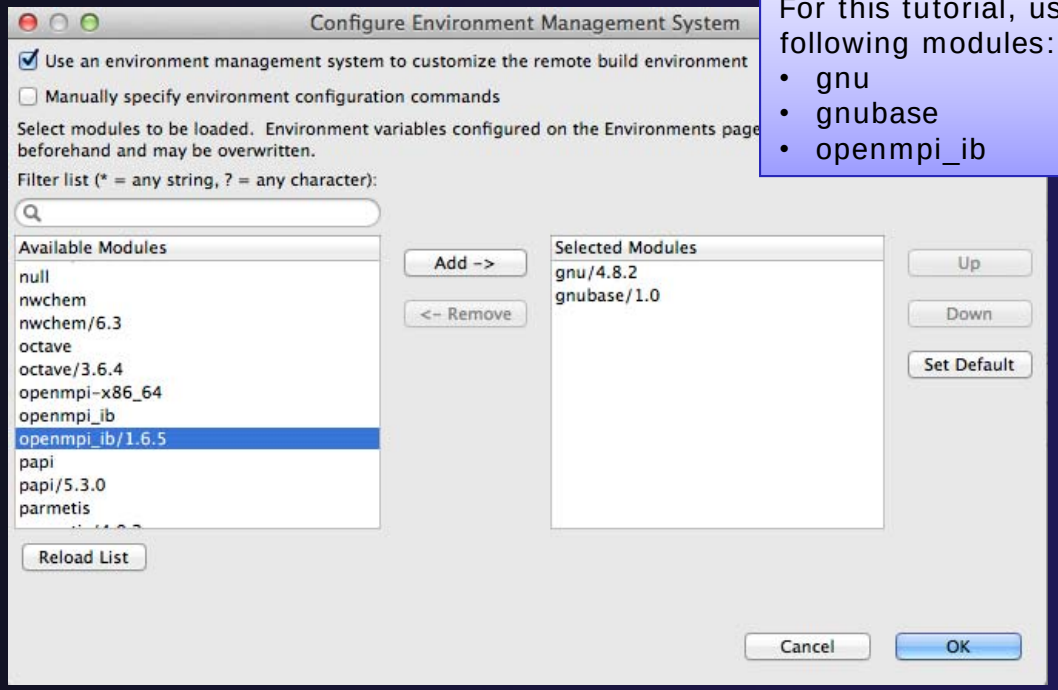
Resources Tab

- 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*



Configure Environment Modules

- Click on the *Modules to Load:* **Configure…** button
- Check the *Use an environment management system to customize the remote build environment* box if it is not already checked
- Select the required modules and click **Add ->** (you can either select one at a time, or all at once)
- Click ok

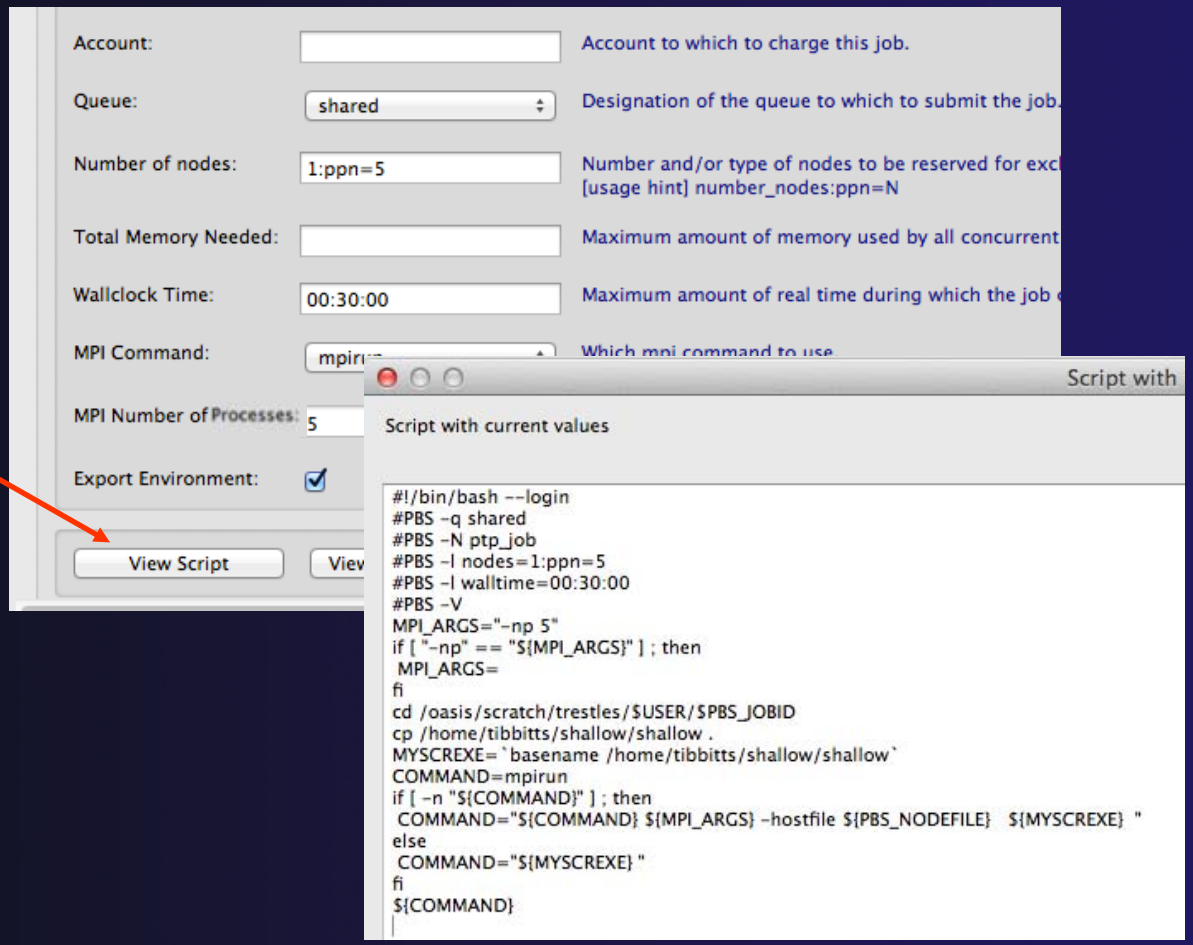


For this tutorial, use the following modules:

- gnu
- gnubase
- openmpi_ib

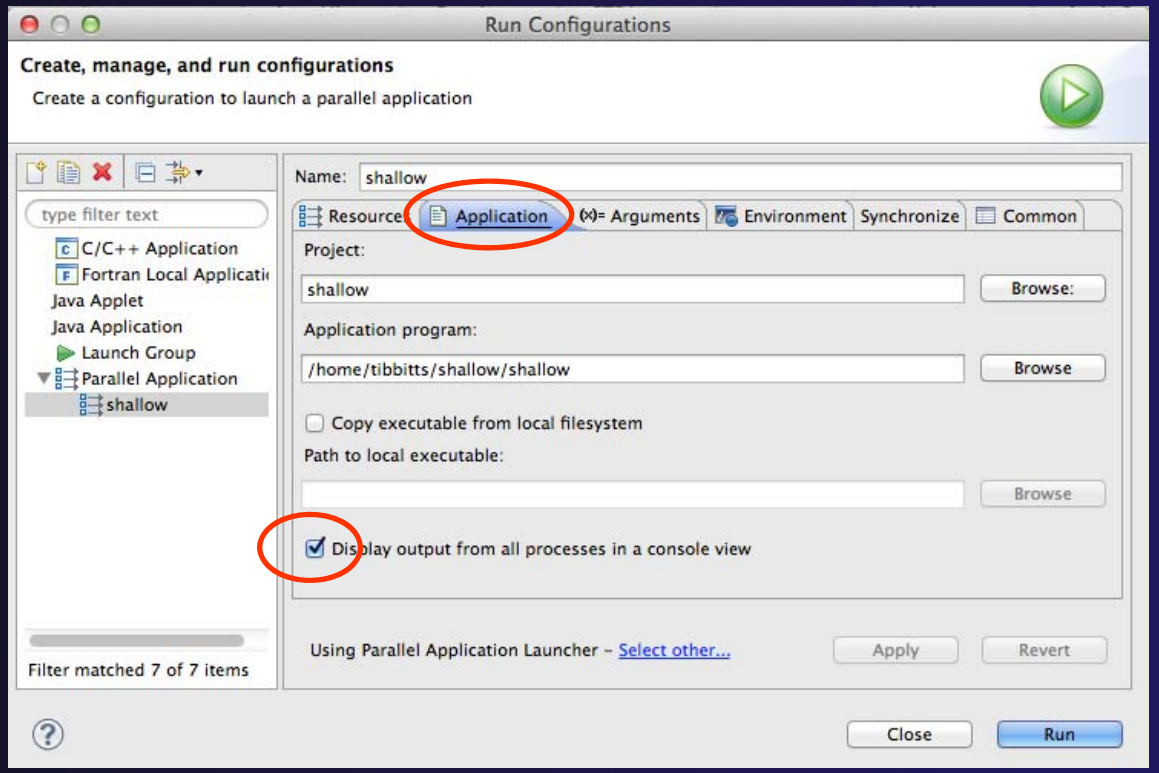
Viewing the Job Script

- 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



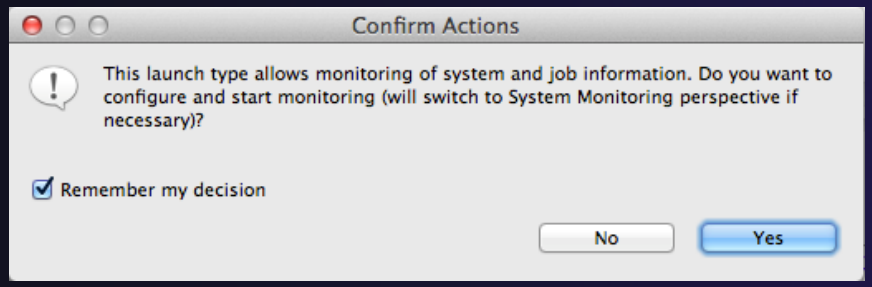
Application Tab

- 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**



Run

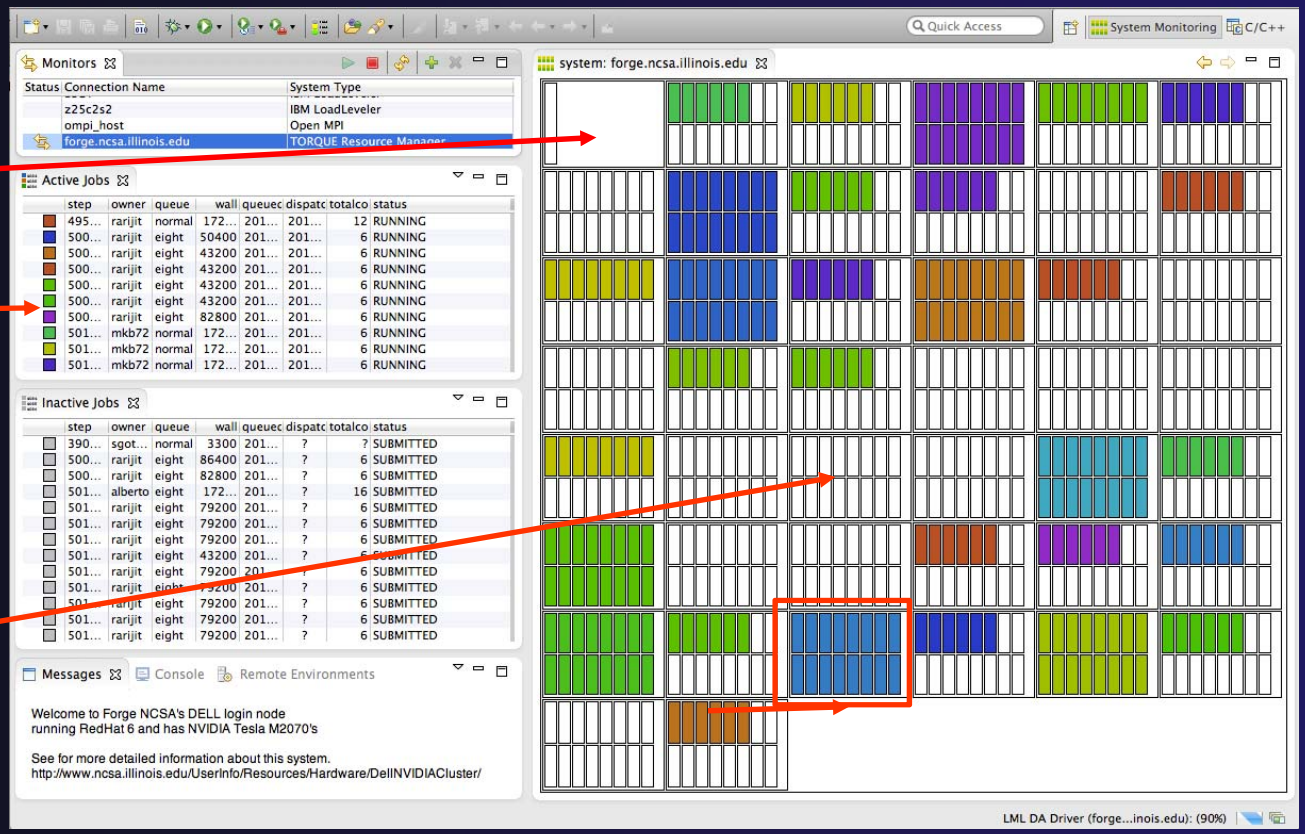
- Select **Run** to launch the job
- You may be asked to switch to the System Monitoring Perspective



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

System Monitoring

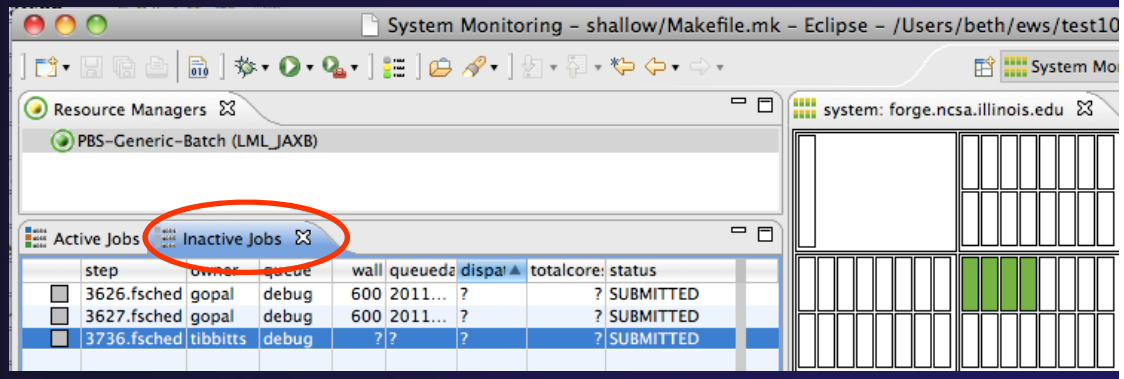
- **System** view, with abstraction of system configuration
- Hold mouse button down on a job in **Active Jobs** view to see where it is running in **System** view
- Hover over node in **System** view to see job running on node in **Active Jobs** view

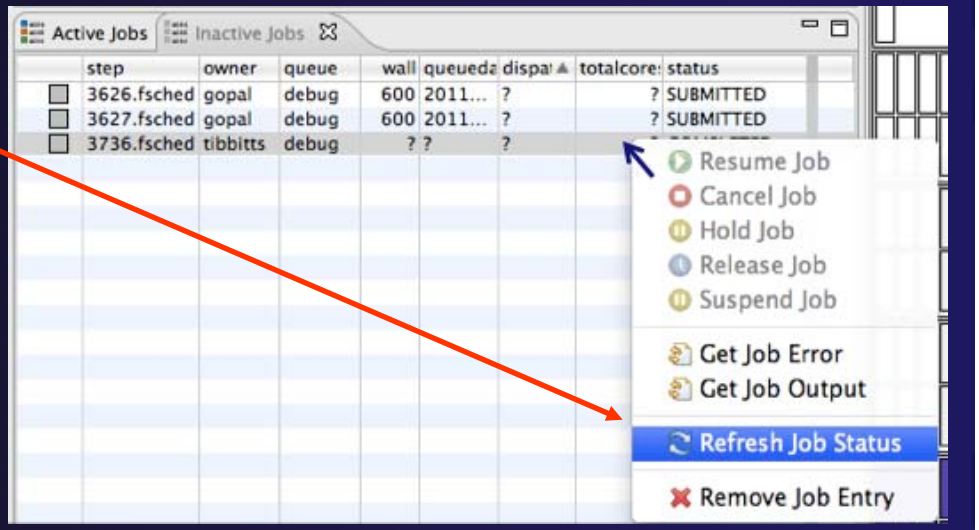


One node with
16 cores

Job Monitoring

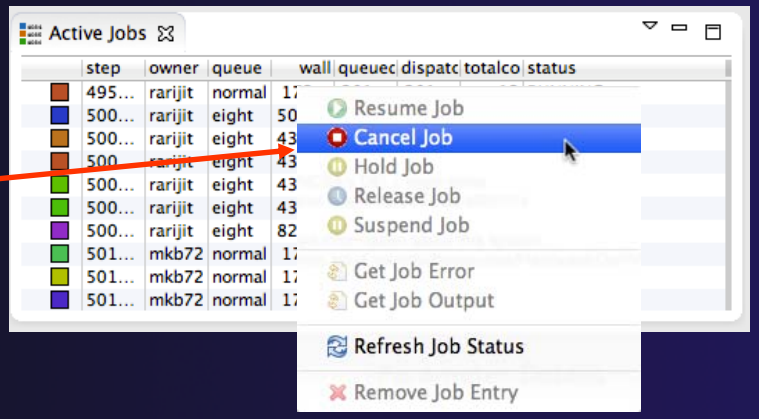
- Job initially appears in **Inactive Jobs** view
- Moves to the **Active Jobs** view when execution begings
- Returns to **Inactive Jobs** view on completion
- Status refreshes automatically every 60 sec
- Can force refresh with menu

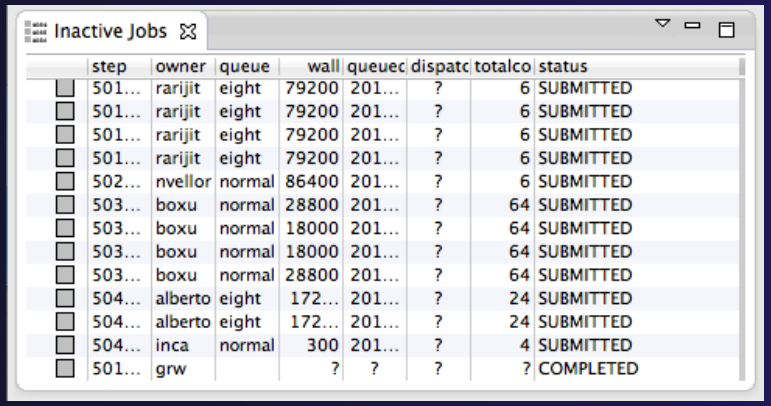




Controlling Jobs

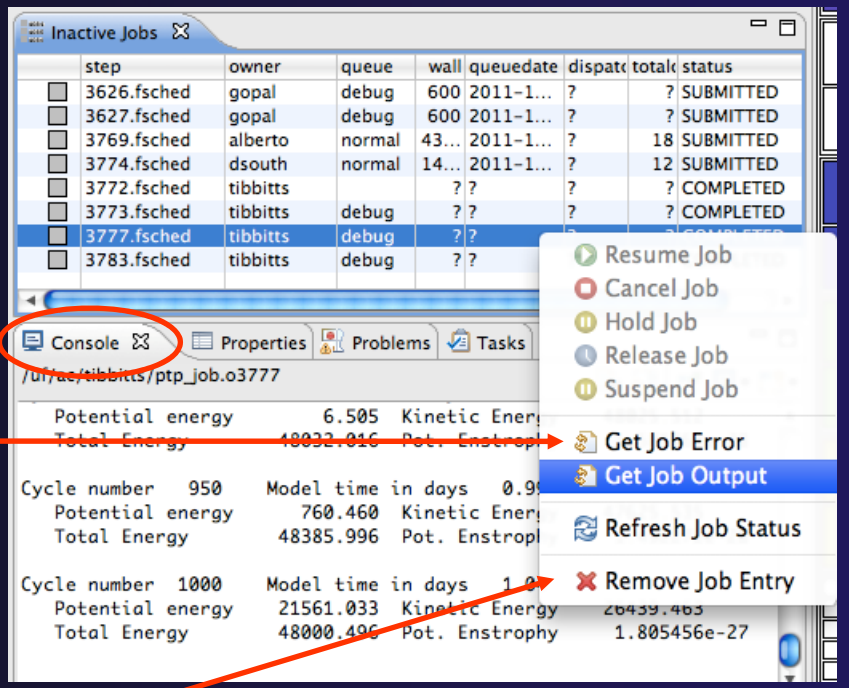
- Right click on a job to open context menu
- Actions will be enabled IFF
  - The job belongs to you
  - The action is available on the target system
  - The job is in the correct state for the action
- When job has COMPLETED, it will remain in the **Inactive Jobs** view





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**



Tutorial 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 (part of Parallel package)
- Python
  - http://pydev.org
- Ruby
  - http://www.aptana.com/products/radrails
- Perl
  - http://www.epic-ide.org
- VI bindings
  - Vrapper (open source) - http://vrapper.sourceforge.net
  - viPlugin (commercial) - http://www.viplugin.com

Online Information

- Information about PTP
  - PTP online help
    - http://help.eclipse.org
  - Main web site for downloads, documentation, etc.
    - http://eclipse.org/ptp
  - Wiki for designs, planning, meetings, etc.
    - http://wiki.eclipse.org/PTP
- Information about Photran
  - Main web site for downloads, documentation, etc.
    - http://eclipse.org/photran