1. Goals
2. Eclipse
   - Installing the Oracle JDK
   - Installing Eclipse
3. Hello World in C
   - Advanced Editing
   - Linux Tools
   - Hello World in C+MPI
   - Adding Message Passing
   - Parallel Analysis Tools
4. Remote Project
   - Resource Managers
   - Adding a Remote Project
   - Run Configuration
   - PBS
5. Synchronized Project
Goals
Goals

The module *Advanced Programming Tools* will teach:

- Eclipse
- Installing Eclipse and the JDK
- Views, Editors, Codes
- Tools for Static Analysis
- Tools for Debugging
- Linux Tools for Eclipse
- HPC Toolkit Plugin
- Mojave

We will use Cactus as an example of an Application Framework.
Eclipse
Eclipse

- IDE: Integrated Development Environment
- Advanced editing capabilities
- Written in Java for multi-platform support
- open-source
- extensible
- Supports C, C++, Fortran, etc. through “plugins”

PTP Tutorial
February 29, 2012
Installing the Oracle JDK
Installing the Oracle JDK

- Goto http://java.sun.com
- Click on “Java SE” under “Top Downloads”
- Click on the “Java Download” button
- Scroll down and get the Java SE 6 JDK (Java Development Kit)
Installing the Oracle JDK

After you run the installer, add the following to your .bashrc

- `export JAVA_HOME=/usr/java/jdk1.6.0_27`
- `export PATH=$JAVA_HOME/bin:$PATH`
- Now source your .bashrc
Installing Eclipse
Installing Eclipse

- Get Eclipse IDE for Parallel Application Developers
- Command: `tar xvf ~/Download/eclipse-parallel-indigo-SR2-incubation-linux-gtk-x86_64.tar.gz`
- Command: `cd eclipse/`
- Command: `./eclipse  -XX:PermSize=256m -XX:MaxPermSize=512m &`
- You’ll see a prompt for selecting a workspace. Check the box which says “Use this as the default and do not ask again”
Hello World in C
Hello World in C

- File > New > C Project
- Select “Hello World ANSI C Project”
- Fill in project name
- Click “Next”
“Project > Build Project” builds a project.
“Run > Run” runs the program
Advanced Editing
Control Sequences

- Find source definition (Ctrl-G to show, F3 to go)
- Alt← navigates to the previous source window
- Shift-Ctrl-G find symbol in workspace
- Ctrl-H to search
Control Sequences

- Alt-Shift-← or Alt-Shift-→ highlights a block
- Correct indentation: highlight region, then Ctrl-I
- Toggle comment: Ctrl-/  
- Correct format: highlight region, then Ctrl-Shift-F
- But what is the correct style?
Project Style

- Select a project
- Select: Project > Properties
- Open C/C++ General
- Select “Code Style”
A perspective is a collection of views. You can change them by going to Window > Open Perspective...

On the right you see the outline view.

Click the small x and it will go away.

To bring it back, use Window > Open View... (or Shift-Alt-Q)
The “History” view tracks session edits (Shift-Alt-Q Z)
It can recover old versions from within the session
By default, you won’t see edits from a previous session, but if you click the “Link with Editor and Selection” you can see them.
Old versions stored in
.metadata/.plugins/org.eclipse.core.resources/.history
Preferences

- Eclipse rebuilds all open projects before running. Very annoying.
- To turn it off, go to Window > Preferences
- Open Run/Debug
- Select Launching
- Uncheck “Build (if required) before launching”
Automatic Updates

- Configure automatic updates
- Go to Window > Preferences
- Go to Install/Update > Automatic Updates
Refactorings

- Alt-Shift-R rename variable or function
- Alt-Shift-L extract local variable
- Alt-Shift-M extract method
- Alt-Shift-Z Surround with...
Linux Tools
LinuxTools

- Help > Install New Software...
- Hit “Next” etc.
Create, manage, and run configurations
Profile C/C++ Application Using Valgrind

Name: Hello Valgrind
Tool to run: Memcheck

General Options

- Child silent after fork
- Run __libc_freeres on exit

Basic Options

- Trace children on exec
- Demangle C++ names

Error Options

Callers in stack trace: 12
LinuxTools: Valgrind

```
char message[100];
printf("Hello from army\n");

MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &p);
```

Hello Debug (1) [memcheck] /usr/bin/valgrind (10/5/11 8:17 PM)

- Invalid read of size 1 [PID: 5917]
  - at 0x4A06302: __GI_strlen (mc_replace_strmem.c:284)
  - by 0x376D66EAA: puts (libc-2.12.2.so)

- Process terminating with default action of signal 11 (SIGSEGV) [PID: 5917]
LinuxTools: OProfile Setup

Step 1:

```
[root@localhost /home/sbrandt/Apps/eclipse/plugins/org.eclipse.linuxtools.oprofile.core_0.3.1.201109262035/natives/linux/scripts]$
[root@localhost scripts]# sh ./install.sh
Eclipse-OProfile plugin install successful.
[root@localhost scripts]$
```

Step 2:

![Profile Configurations Window]

Create, manage, and run configurations

- **Type filter text**
- **C/C++ Application**: Hello (1)
- **Launch Group**
- **Parallel Application**: HelloPar
- **Profile With OProfile**: Hello Debug
- **Profile With OProfile**: Profile with Start

**Name:** Hello Debug

**C/C++ Application:**
- **Debug/Hello**

**Project:**
- **Hello**

**Build (if required) before launching**

**Build configuration:** Debug

- **Apply**
- **Revert**

- **Close**
- **Profile**
The result of running oprofile is that the oprofile view gives you information about your run. You can click inside the view and see the relevant line in the editor.
Select Project and right click

Project Properties > C/C++ Build > Settings > Debugging

Click on “Generate gprof information (-pg)”

Compile and run

Highlight project and hit F5 (refresh)

Linux Tools can process gmon.out
Hello World in C+MPI
Hello World in C+MPI

Upgrading the program to use MPI

- Add #include <mpi.h>
- Click inside the program type “mpi” then hit Ctrl-space. You’ll see a code completion options. Choose “MPI Init and Finalize” you now have errors in your code. Use the editor to add argc and argv to main.
Hello World in C+MPI

Building will now produce errors.
Hello World in C+MPI

To resolve these errors...

- Right click on the “Hello” project in the project explorer view.
- Select “Properties”
- Open “C/C++ General” and click on “Paths and Symbols”
- Select the C language and click “Add” to add the include.

![Image of Visual Studio interface showing project properties and include directory configuration]
How did I know the mpi include path?

- Run `sh -x mpicc test.c`
- Output will contain something like this: 
  
  ```
  + gcc -m64 -O2 -fPIC -Wl,-z,noexecstack test.c -l/usr/include/mpich2-x86_64 -L/usr/lib64/mpich2/lib -L/usr/lib64/mpich2/lib -lmpich -lomp -lpthread -lrt
  ```
- Now you can see the include (`/usr/include/mpich2-x86_64`), the lib dir (`/usr/lib64/mpich2/lib`), libs, etc.
While you’re still in “C/C++ General”, click on the “Library Paths” tab and fix that. (In my case, add /usr/lib64/mpich2/lib)

Next, go to the “Libraries” tab and fix that.
“Project > Build” should now work again.

Now we need to set up a parallel run. Click “Window > Open Perspective > Other… > Parallel Runtime”

Right click inside the “Resource Managers” tab.

Click on “Add Resource Manager…”

Choose either MPICH2 or OpenMP, whichever you have on your machine.

Click Next > Finish.

In a separate window, type mpdboot to start mpich2

Right click on the Resource manager and select start.
Hello World in C+MPI

- Click Run > Run Configurations...
- Right Click on Parallel Applications and select New
- Select the new application, configure the “Application program.”
- Click “Apply” and “Run”
Hello World in C+MPI

- Click Run > Run Configurations...
- Select your parallel application
- Select the “Resources” tab
- Adjust the “Number of processes” to 2
- Click “Apply” and “Run”
Adding Message Passing
Hello World in C+MPI

- Navigate back to the C/C++ perspective (Window > Open Perspective > C/C++)
- Click in the code editor between MPI_Init and MPI_Finalize.
- Type “mpi” and hit Ctrl-space. Take the “mpisr” code completion. A complete skeleton for doing an MPI send and receive will appear. Fill in the missing variable declarations.
  ```
  int rank,p,source,dest,tag=66;
  MPI_Status status;
  char message[100];
  ```
- Alt-P followed by B will build the current project.
- Click on the down arrow and select your run config.
Parallel Analysis Tools
Using Parallel Analysis

- The most basic form of parallel analysis is available through the parallel analysis button.
- Select a source file in the package explorer.
- Click the down arrow, then choose “Show MPI Artifacts”.
Using Parallel Analysis

- Select a source file in the package explorer
- Click the down arrow, then choose “MPI Barrier Analysis”
Using Parallel Analysis

- Now introduce a barrier error
- Select a source file in the package explorer
- Click the down arrow, then choose “MPI Barrier Analysis”
Using GEM

- To use the GEM tools, you must first install isp
- Isp is a tool from University of Utah to analyze C-language (not C++) programs and discover MPI errors.
- It contains a compiler (ispcc), a tool to run a program (isp), and a viewer. This tool may be used outside of Eclipse.
- Installation of isp follows the familiar pattern of configure / make / make install. Make sure the ispcc command is in your path when you start Eclipse.
- Download page: http://www.cs.utah.edu/formal_verification/ISP-release/
Using GEM

- To run GEM in Eclipse, click the trident.
- You will be prompted for command line arguments. Click “OK”.
- GEM will now analyze your code. Click the “GEM Analyzer” tab, then “Browse MPI Calls.”
Now let’s introduce an error. A call to MPI_Barrier() just before the call to MPI_Send().

Save the file. Build the project. Run GEM. Deadlock detected.
Remote Project
To change perspective, go to \textit{Window} $\rightarrow$ \textit{Open Perspective} $\rightarrow$ \textit{Other}...
Select *Parallel Runtime*. 
The perspective is now open.
Right-click in the white space under the *Resource Managers* tab and select the *Add Resource Manager* menu item.
Add a *Remote Launch*. 
Select *Remote Tools* for *Remote service provider* and click *New* to configure a new connection.
For the connection configuration, enter the hostname and credentials. Click *Finish* until finished.
Back in the parallel runtime perspective, right-click the new RM and select **Start Resource Manager**.
You may be asked if you’d like to continue connecting; hit Yes.
The RM is now started.
Remote Project
File → New → Project...
Select *Remote C/C++ Project.*
Configure as shown below, then click *Finish*. 
Create a new source file by right-clicking on the project name, then **New → Source File**.
Name the file.
Type a simple Hello, World program into the editor.
Create a new makefile by right-clicking on the project name, then New → File. Fill in the makefile as follows.

```
all:
gcc hello.c -o hello

clean:
rm hello
```
Remote Project
Before you attempt to run, you need to *Build* using the hammer symbol.
Click on the drop-down arrow next to the green arrow and click *Run Configurations*...
Click the blank page with the plus sign to create a new configuration. Set the resource manager to the one you created earlier.
Select the *Application* tab to complete the configuration. For *Application program*, hit browse and select the location on the remote machine where the executable will reside.
Hit *Run*. The application will run remotely.
Follow the steps for creating a new resource manager, except select **PBS-Generic-Batch** this time.
To get a parallel Hello, World code to execute remotely, open another project...
Let’s make an MPI Hello World project and copy the code and makefile.
Once the project is created, right-click on the source and select **Copy**, then paste it under the existing remote launch project.
Do the same for the makefile.
Edit the makefile.
Create a corresponding run configuration for the project, except specify this as the new resource manager.
Under the *Application* tab, fill in the fields.

![Image of Eclipse IDE with configuration settings for a parallel application]

- **Account**: Enter the account to which to charge this job.
- **Queue**: Select the queue to which to submit the job.
- **Number of nodes**: Specify the number of nodes to be used.
- **Total Memory Needed**: Enter the maximum amount of memory used by a process.
- **Wallclock Time**: Specify the maximum amount of real time during which the job will run.
- **MPI Command**: Choose which MPI command to use.
- **MPI Number of Cores**: Specify the number of cores to use.
Hit *Run*. Then, switch to the *System Monitoring* perspective, right-click on the now-inactive job, and click *Show output*. The output will be shown.
Project Types

- **Local Project**: Source is located on local machine and build happens locally
- **Remote**: Source is located on remote machine and build happens remotely
- **Synchronized** = **Local** + **Remote**
Remote Project Process Layout
Synchronized Project Layout

Synchronized Projects

Local source code

Remote

Synchronize

Search/Index Navigation

Build Service

Debug Service

Launch Service

Run

Debug

Build

Executable

File Service

Index Service

Edit

Local

Source code copy

LSU
Demo

- Checkout a project from SVN.
- Building a default Makefile project.
- Building a custom Makefile Project by changing the command to build application.
- Launching the application locally.
- Converting the project into synchronized project.
- Building and launching the project remotely using the cluster resource manager.
Select File → New → Others
Checkout a project from SVN.

https://svn.cct.lsu.edu/repos/cactus/ppa-mojave/mojave/demo
Checkout files in repo as C project.
Select empty Makefile project in Wizard.
Building project with custom command.

Right click on the project and select properties
Launch MPI-Based application on local machine.

- Windows → Open Perspective → Others → System Monitoring.
- In resource manager, right click → add resource manager → MPICH2.
- Select Local for remote service provider and connection name.
- Start resource manager by right click and click start.
System Monitoring view.
Steps to launch application.

- Click Run in main menu → Run configuration → Parallel application (right click) → new → name it and choose MPICH2 as resource manager (choose number of process in multiple of 4).
- In application tab, select the project and choose executable.
- Hit run.
- Console shows the output for locally running MPI program.
Converting Local/Team project into Synchronized project

File → New → Others → Remote → convert C/C++ or Fortran project to a Synchronized project
Converting Local/Team project into Synchronized project

- Click remote tools for remote service provider.
- Icon changes from repo (yellow cylinder) to synchronized (double headed arrow).
- All synchronized projects saves all files before buid and once user saves it locally.
- To sync manually right click on project → synchronization → sync all now.
- To build remotely, right click on project → Build Configuration → Set Active → Remote. Then hit the hammer to build the application on remote machine.
Launch job using PBS on remote machine

In System monitoring perspective, right click on resource manager view → add resource manager → PBS-Generic-Batch and proceed with remote tools on Queenbee machine and start the RM.
Create a run configuration

- Run → run configurations. Create new configuration for the project under parallel application.
- Select PBS-Generic-Batch for resource manager combo-box.
- Fill account with your allocation name and the queue.
- In number of nodes type number: ppn=8
- In application tab pick the executable from the Queenbee location and hit run.
Executable from QueenBee.
Sample Run config.
Launch and monitor job

Inactive jobs view shows job is submitted.
Job moves to Active job view when it is running and right click on job and hit job status to refresh the status of the job.
When the job is completed, it will be shown in inactive jobs view. By right click on the jobs in inactive job, view output/error by clicking get job output/error option.