# XSEDE Training Resources

John Lockman
Advanced Scientific Computing Training Coordinator

XSEDE
Extreme Science and Engineering Discovery Environment

NSF

# XSEDE Training Mission

To develop and enhance the skills of the national open science community for the effective conduct of research and education activities utilizing advanced computational resources.

# Training Focus

- Systems and Software supported by XSEDE Service Providers
  - Systems: Stampede, Blacklight, Gordon, and others
- Principles and techniques for effective use of resources
  - Programming
  - Program optimization
  - Data management

# Training Opportunities

- In Person
- Webcast
- Self paced Learning
- On-site – Bring Your Own Code Days

# In Person Training

National Center for Supercomputing Applications, Urbana-Champaign, IL

- https://bluewaters.ncsa.illinois.edu/training

Texas Advanced Computing Center, Austin TX

- https://www.tacc.utexas.edu/user-services/training

Pittsburgh Supercomputing Center, Pittsburgh PA

- http://www.psc.edu/index.php/users/training

National Institute for Computational Sciences, Knoxville TN

- http://www.nics.tennessee.edu/hpc-seminar-series

San Diego Supercomputing Center, San Diego CA

- http://www.sdsc.edu/us/training/

# Example Courses

## Introduction Courses:

Linux Basics

Introduction to C & Fortran

Introduction to Parallel Computing

## Intermediate / Advanced Courses:

Optimizing Code for Xeon Phi

Parallel I/O

HPC for NextGen Sequence Analysis

# Self Paced Learning

Cornell Virtual Workshop

https://www.cac.cornell.edu/VW/

CI Tutor

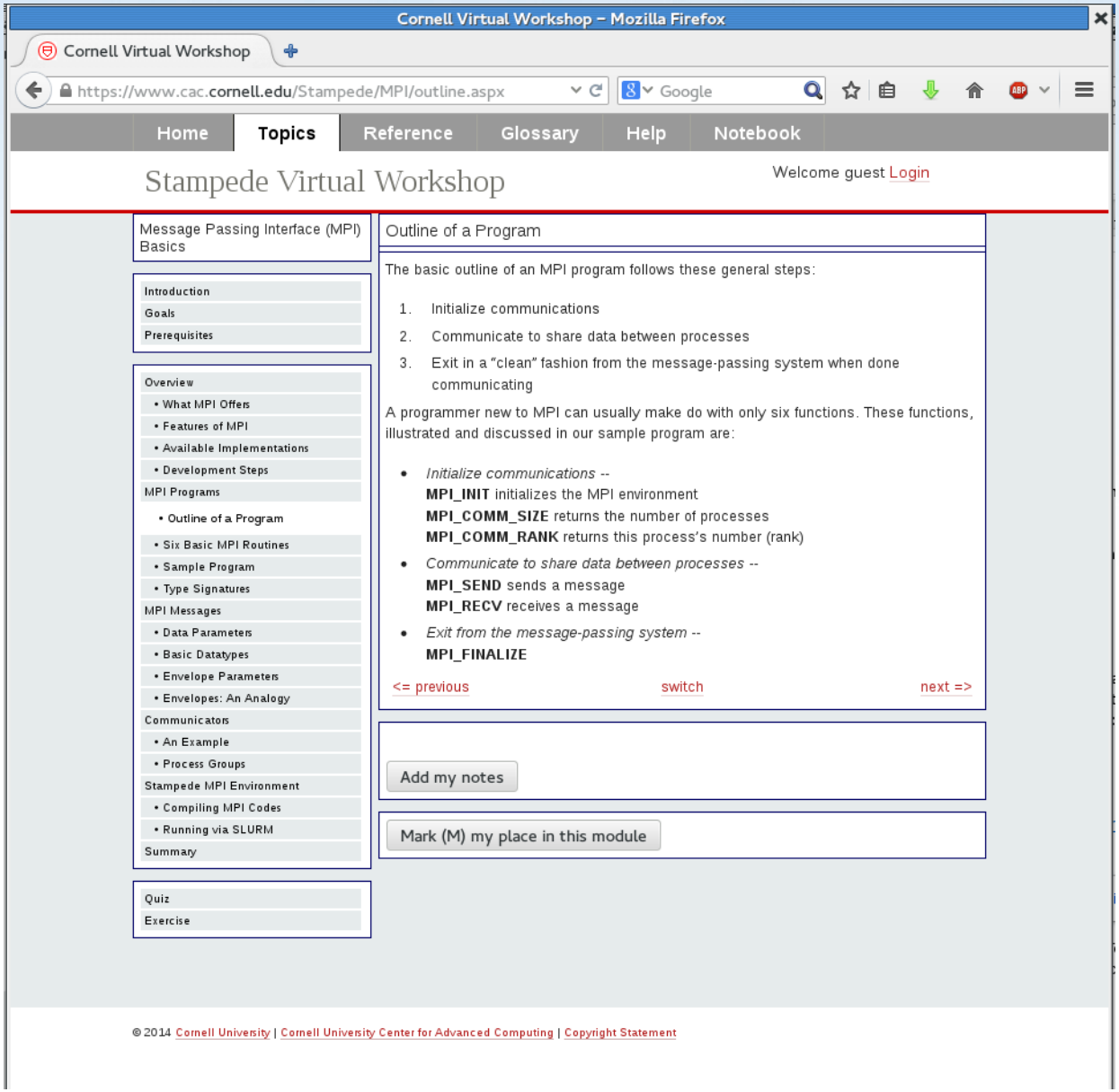
http://www.citutor.org

HPC University

http://www.hpcuniversity.org/

# Cornell Virtual Workshop

- High Performance Computing topics:
  - system architecture
  - parallel programming with MPI and OpenMP
  - code optimization for performance and scalability
  - parallel code debugging
  - basic visualization
  - data management


Cornell Virtual Workshop – Mozilla Firefox
Cornell Virtual Workshop
https://www.cac.cornell.edu/Stampede/MPI/outline.aspx
Google
Home
Topics
Reference
Glossary
Help
Notebook
Stampede Virtual Workshop
Welcome guest Login
Message Passing Interface (MPI) Basics
Introduction
Goals
Prerequisites
Overview
• What MPI Offers
• Features of MPI
• Available Implementations
• Development Steps
MPI Programs
• Outline of a Program
• Six Basic MPI Routines
• Sample Program
• Type Signatures
MPI Messages
• Data Parameters
• Basic Datatypes
• Envelope Parameters
• Envelopes: An Analogy
Communicators
• An Example
• Process Groups
Stampede MPI Environment
• Compiling MPI Codes
• Running via SLURM
Summary
Quiz
Exercise
Outline of a Program
The basic outline of an MPI program follows these general steps:
1. Initialize communications
2. Communicate to share data between processes
3. Exit in a "clean" fashion from the message-passing system when done communicating
A programmer new to MPI can usually make do with only six functions. These functions, illustrated and discussed in our sample program are:
Initialize communications --
MPI_INIT initializes the MPI environment
MPI_COMM_SIZE returns the number of processes
MPI_COMM_RANK returns this process's number (rank)
Communicate to share data between processes --
MPI_SEND sends a message
MPI_RECV receives a message
Exit from the message-passing system --
MPI_FINALIZE
<= previous
switch
next =>
Add my notes
Mark (M) my place in this module
© 2014 Cornell University | Cornell University Center for Advanced Computing | Copyright Statement

Cornell Virtual Workshop – Mozilla Firefox

Cornell Virtual Workshop

https://www.cac.cornell.edu/Stampede/MPI/exercise.aspx

Google

Home | Topics | Reference | Glossary | Help | Notebook

# Stampede Virtual Workshop

Welcome guest Login



## Exercises

**Make sure MPI is working**

The first thing to do is to ensure that you're actually able to compile and run MPI code. In order to do this, just grab the "Hello World" code in your preferred language (C or Fortran), and paste it into your favorite editor on Stampede. Compile it using your compiler of choice, e.g.:

```
login3% mpicc -o mpi_silly -O2 hello_mpi.c
```

Drop the binary into your home directory, and adjust your batch script to execute the app. To start with, we'll just run 16 MPI tasks to fill up 1 Stampede node (16 cores).

```
#!/bin/csh

#SBATCH -J myMPI          # Job Name
#SBATCH -o myMPI.o%j      # Name of the output file
(myMPI.oJobID)
#SBATCH -p development    # Queue name
#SBATCH -t 00:05:00       # Run time (hh:mm:ss) - 5 minutes
#SBATCH -N 1              # Requests 1 MPI node
#SBATCH -n 16             # 16 tasks total
#SBATCH -e myMPI.err%j    # Direct error to the error file
#SBATCH -A TG-TRA120006   # Account number

ibrun mpi_silly
```

Go ahead and submit the job, and verify that the contents of the stdout look something like this:

```
TACC: Starting up job 562026
TACC: Setting up parallel environment for MVAPICH2+mpispawn.
TACC: Starting parallel tasks...
Message from process = 6 : Hello, world
Message from process = 5 : Hello, world
Message from process = 2 : Hello, world
Message from process = 3 : Hello, world
Message from process = 10 : Hello, world
...
```

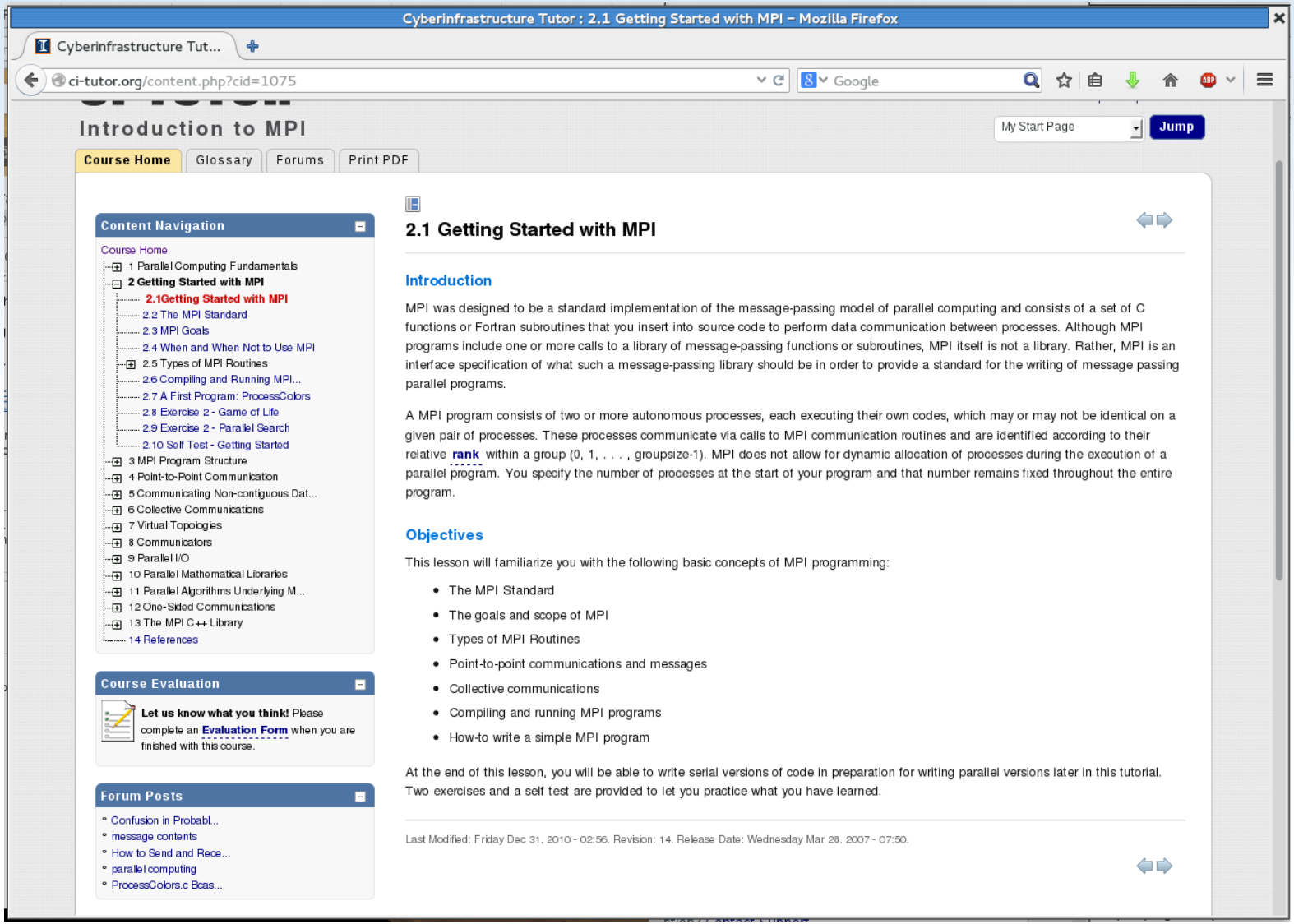
The output simply shows all of the workers giving their id and repeating the message from the master saying hello. At this point, we can abandon Hello World programs and play with something slightly more interesting.

**Divide and conquer**

There is a class of problems that are relatively easy to parallelize where we have a number of identical operations that need to be performed and some accumulation function operates on the individual outputs. A good example is integration of some function, where we'll be determining the area of many small pieces and summing them to get the total area. The more small integrations we will do the better an approximation of the total area we will get, so there is an advantage to calculating many small sums. The two common ways to calculate the small sums are the Trapezoidal Rule and Simpson's Rule. Simpson's is a little more general, so let's start with that. We'll start with a serial

# Cyberinfrastructure Tutor (CI Tutor)

- Tutorials on high-end computing topics
  - parallel computing
  - multi-core performance
  - performance tools

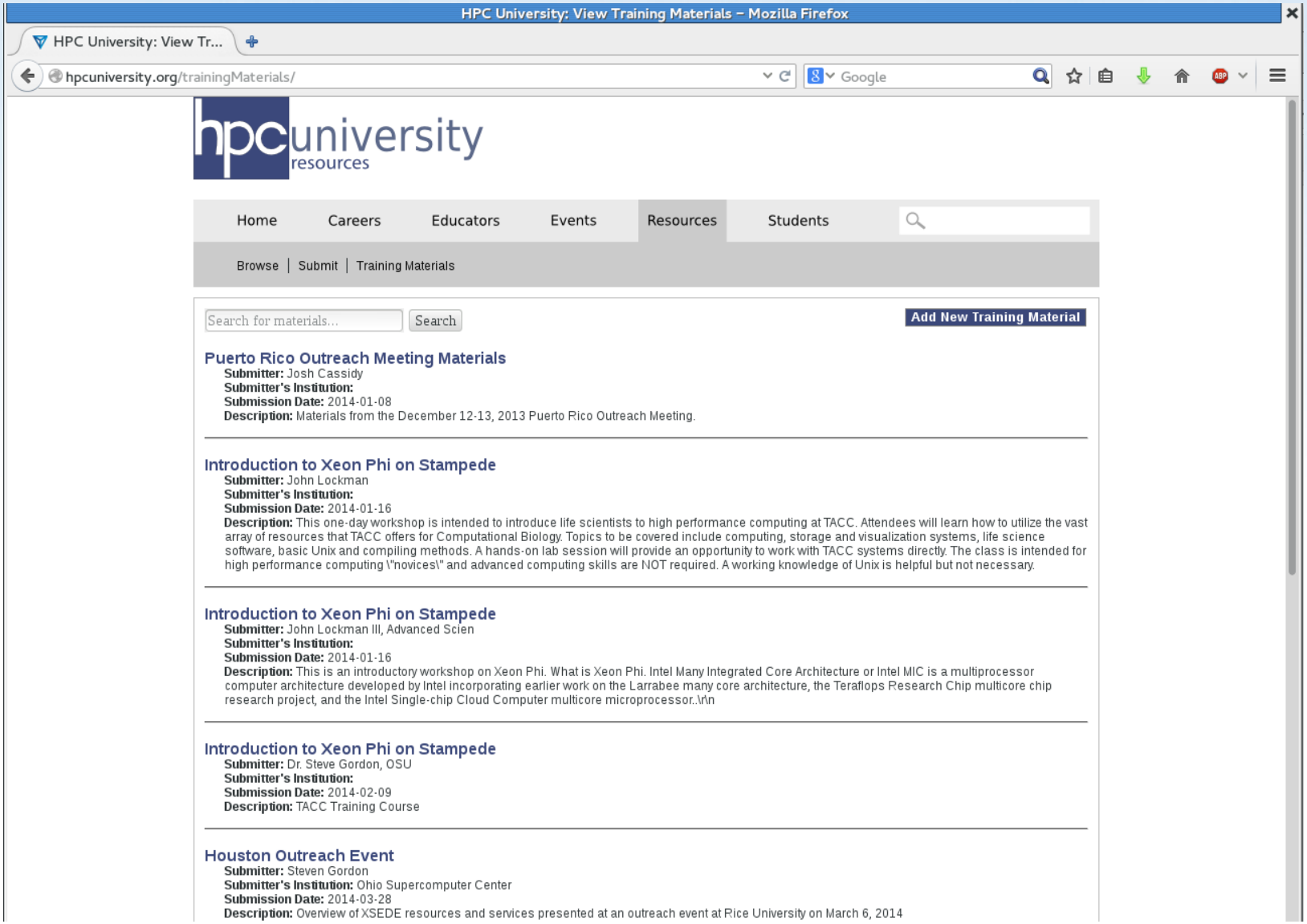

Cyberinfrastructure Tutor : 2.1 Getting Started with MPI – Mozilla Firefox
Cyberinfrastructure Tut...
ci-tutor.org/content.php?cid=1075
Google
Introduction to MPI
My Start Page
Jump
Course Home
Glossary
Forums
Print PDF
Content Navigation
Course Home
1 Parallel Computing Fundamentals
2 Getting Started with MPI
2.1Getting Started with MPI
2.2 The MPI Standard
2.3 MPI Goals
2.4 When and When Not to Use MPI
2.5 Types of MPI Routines
2.6 Compiling and Running MPI...
2.7 A First Program: ProcessColors
2.8 Exercise 2 - Game of Life
2.9 Exercise 2 - Parallel Search
2.10 Self Test - Getting Started
3 MPI Program Structure
4 Point-to-Point Communication
5 Communicating Non-contiguous Dat...
6 Collective Communications
7 Virtual Topologies
8 Communicators
9 Parallel I/O
10 Parallel Mathematical Libraries
11 Parallel Algorithms Underlying M...
12 One-Sided Communications
13 The MPI C++ Library
14 References
Course Evaluation
Let us know what you think! Please complete an Evaluation Form when you are finished with this course.
Forum Posts
Confusion in Probabl...
message contents
How to Send and Rece...
parallel computing
ProcessColors.c Bcas...
2.1 Getting Started with MPI
Introduction
MPI was designed to be a standard implementation of the message-passing model of parallel computing and consists of a set of C functions or Fortran subroutines that you insert into source code to perform data communication between processes. Although MPI programs include one or more calls to a library of message-passing functions or subroutines, MPI itself is not a library. Rather, MPI is an interface specification of what such a message-passing library should be in order to provide a standard for the writing of message passing parallel programs.
A MPI program consists of two or more autonomous processes, each executing their own codes, which may or may not be identical on a given pair of processes. These processes communicate via calls to MPI communication routines and are identified according to their relative rank within a group (0, 1, . . . , groupsize-1). MPI does not allow for dynamic allocation of processes during the execution of a parallel program. You specify the number of processes at the start of your program and that number remains fixed throughout the entire program.
Objectives
This lesson will familiarize you with the following basic concepts of MPI programming:
The MPI Standard
The goals and scope of MPI
Types of MPI Routines
Point-to-point communications and messages
Collective communications
Compiling and running MPI programs
How-to write a simple MPI program
At the end of this lesson, you will be able to write serial versions of code in preparation for writing parallel versions later in this tutorial. Two exercises and a self test are provided to let you practice what you have learned.
Last Modified: Friday Dec 31, 2010 - 02:56. Revision: 14. Release Date: Wednesday Mar 28, 2007 - 07:50.

# HPC University

Provides a cohesive, persistent, and sustainable on-line environment to share educational and training materials

Guides you to:

1) choose successful paths for HPC learning and workforce development
2) contribute high-quality and pedagogically effective materials that allow individuals at all levels and in all fields of study to advance scientific discovery


HPC University: View Training Materials – Mozilla Firefox
HPC University: View Tr...
hpcuniversity.org/trainingMaterials/
Google
hpcuniversity resources
Home
Careers
Educators
Events
Resources
Students
Browse | Submit | Training Materials
Search for materials...
Search
Add New Training Material
Puerto Rico Outreach Meeting Materials
Submitter: Josh Cassidy
Submitter's Institution:
Submission Date: 2014-01-08
Description: Materials from the December 12-13, 2013 Puerto Rico Outreach Meeting.
Introduction to Xeon Phi on Stampede
Submitter: John Lockman
Submitter's Institution:
Submission Date: 2014-01-16
Description: This one-day workshop is intended to introduce life scientists to high performance computing at TACC. Attendees will learn how to utilize the vast array of resources that TACC offers for Computational Biology. Topics to be covered include computing, storage and visualization systems, life science software, basic Unix and compiling methods. A hands-on lab session will provide an opportunity to work with TACC systems directly. The class is intended for high performance computing \"novices\" and advanced computing skills are NOT required. A working knowledge of Unix is helpful but not necessary.
Introduction to Xeon Phi on Stampede
Submitter: John Lockman III, Advanced Scien
Submitter's Institution:
Submission Date: 2014-01-16
Description: This is an introductory workshop on Xeon Phi. What is Xeon Phi. Intel Many Integrated Core Architecture or Intel MIC is a multiprocessor computer architecture developed by Intel incorporating earlier work on the Larrabee many core architecture, the Teraflops Research Chip multicore chip research project, and the Intel Single-chip Cloud Computer multicore microprocessor.\r\n
Introduction to Xeon Phi on Stampede
Submitter: Dr. Steve Gordon, OSU
Submitter's Institution:
Submission Date: 2014-02-09
Description: TACC Training Course
Houston Outreach Event
Submitter: Steven Gordon
Submitter's Institution: Ohio Supercomputer Center
Submission Date: 2014-03-28
Description: Overview of XSEDE resources and services presented at an outreach event at Rice University on March 6, 2014

# More Information

Check out the course calendar:

http://portal.xsede.org/course-calendar

Upcoming Dates

Registration

Course Descriptions