CS/Math 471: Intro. to Scientific Computing

Getting Started with High Performance Computing

Matthew Fricke, PhD

Center for Advanced Research Computing

Table of contents

The Center for Advanced Research Computing

What is CARC

CARC is the hub of computational research at UNM.
We provide HPC resource and consultation services to UNM departments.

Current Projects

Fluid-dynamics

Genetic analysis and evolutionary tree generation of turbuculosis, birds, porcupines.

Structural analysis and design of race cars

Flu dynamics simulations

Swarm robotics simulations

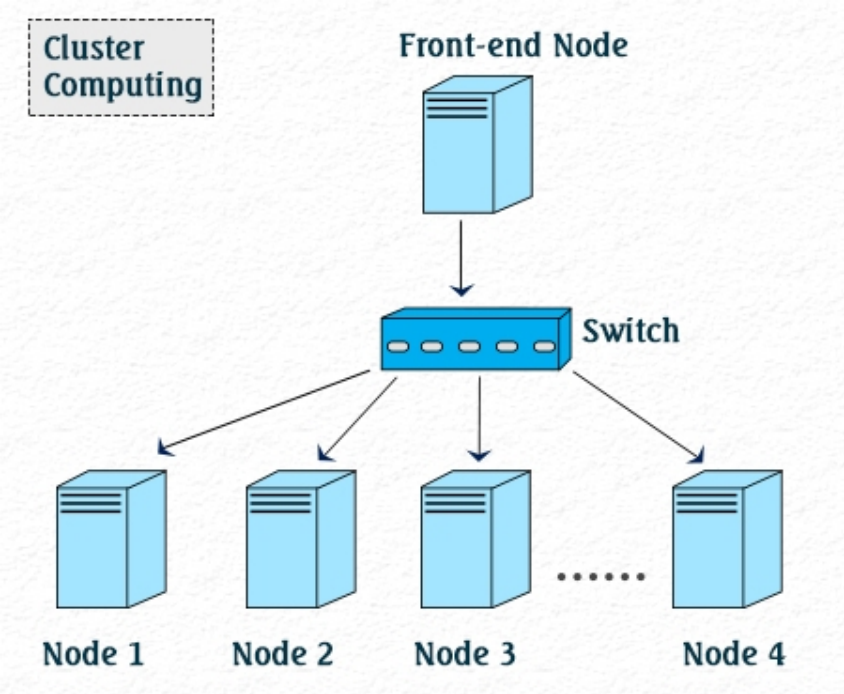
Protein docking

Teaching neural networks to control self-driving cars

Electrodynamic simulations for novel materials

High Performance Clusters (and Wheeler)

Multi-Core



Multiple-compute cores on the same ”node”. Almost all computers you buy today have mutiple cores on one motherboard (dual core through hex-core desktop computers are common). They have shared access to main RAM. For example one of the compute nodes we have on the Taos cluster has 60 cores sharing 3 TB of RAM.

Distributed

Clusters with compute nodes that communicate through a network interface are **distributed**.

In practice, for most compute clusters you will have a distributed system of nodes where each node is itself a multi-core computer.

Wheeler Cluster

296 compute nodes
8 cores per node
48 GB RAM per node
total of 2368 CPUs

File systems

Home directory. Limited space. Limited speed. Store your code here. Backed up. Path: ~

Scratch. Fast. Lot's of space. Not-backed up. Store data that you can regenerate here.
Path: **~/wheeler_scratch**

Temp (on Wheeler these are RAM drives). Very fast but decreases available RAM. Path: **/tmp**

Example Problem: Calculate π

Serial Calculation of π

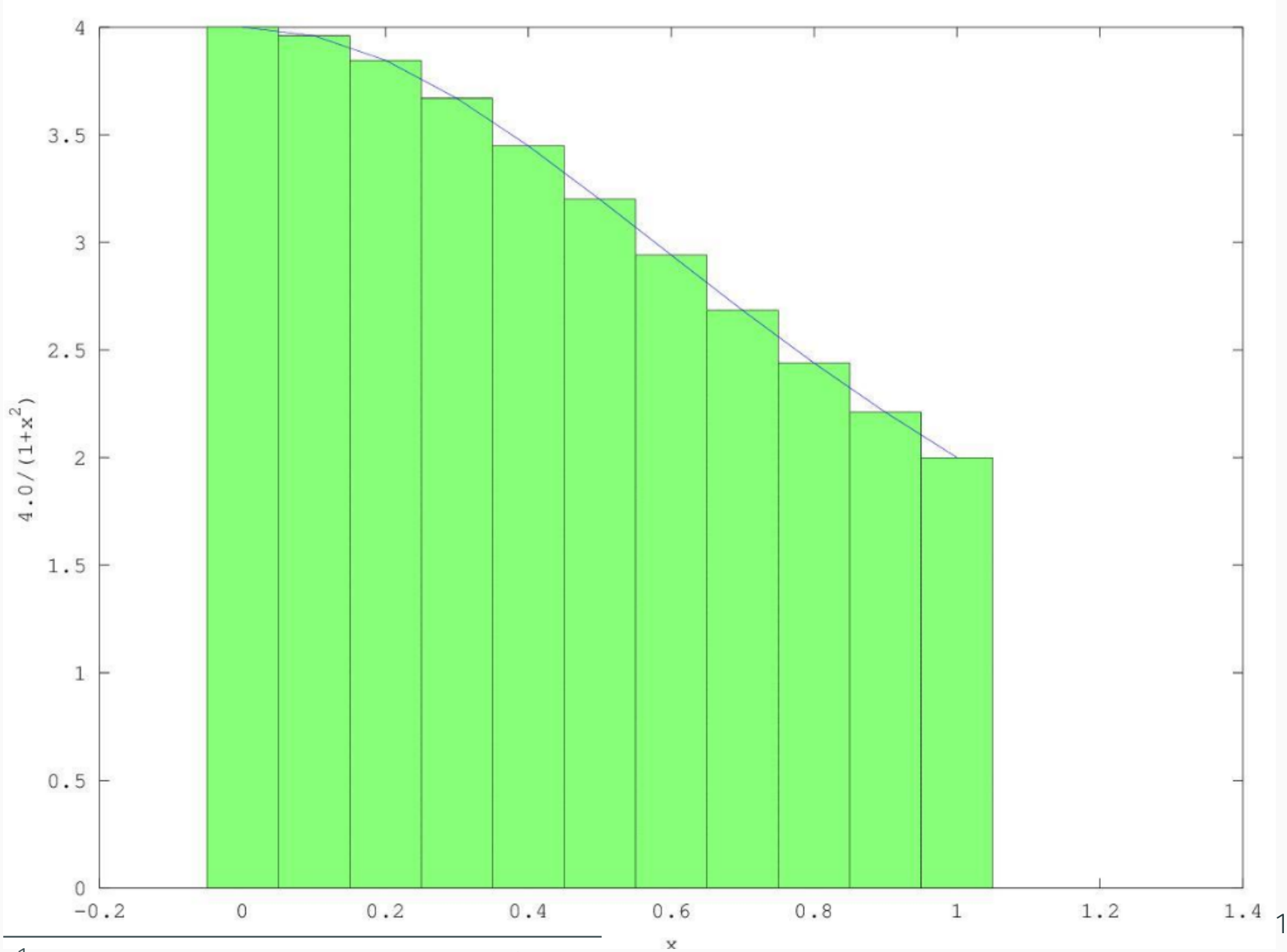
$$\int_0^1 \frac{4}{1+x^2} dx = \pi$$

And can be numerically approximated with:

$$\sum_{i=0}^{N} \frac{4}{1+x_i^2} \Delta x \approx \pi$$

As Δx gets smaller and N gets larger the approximation converges on π

Serial Program to Calculate π



[1]

[1]Sung Bae, Ph.D, New Zealand eScience Infrastructure

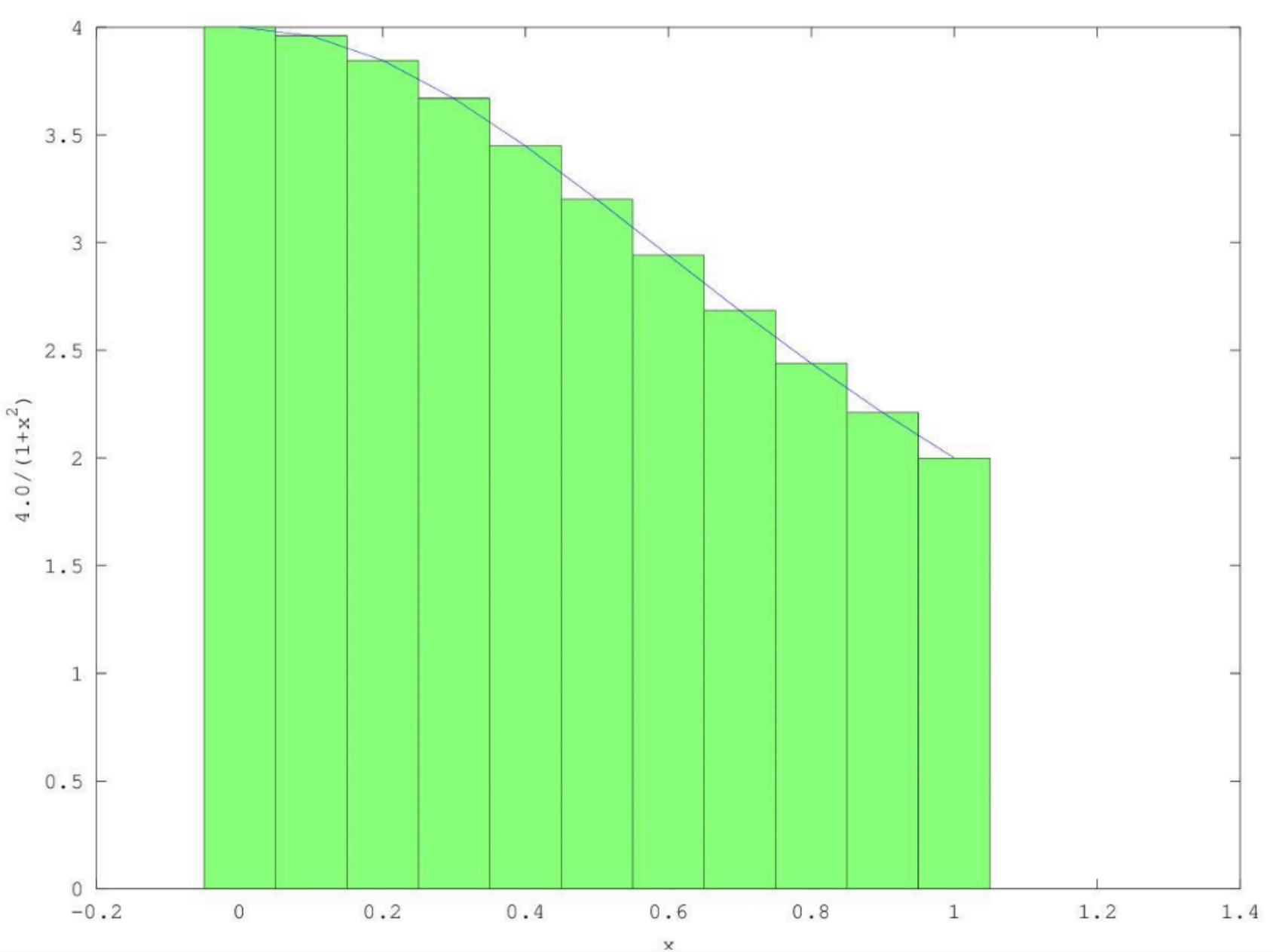
Serial Program to Calculate π

calcPiSerial.py

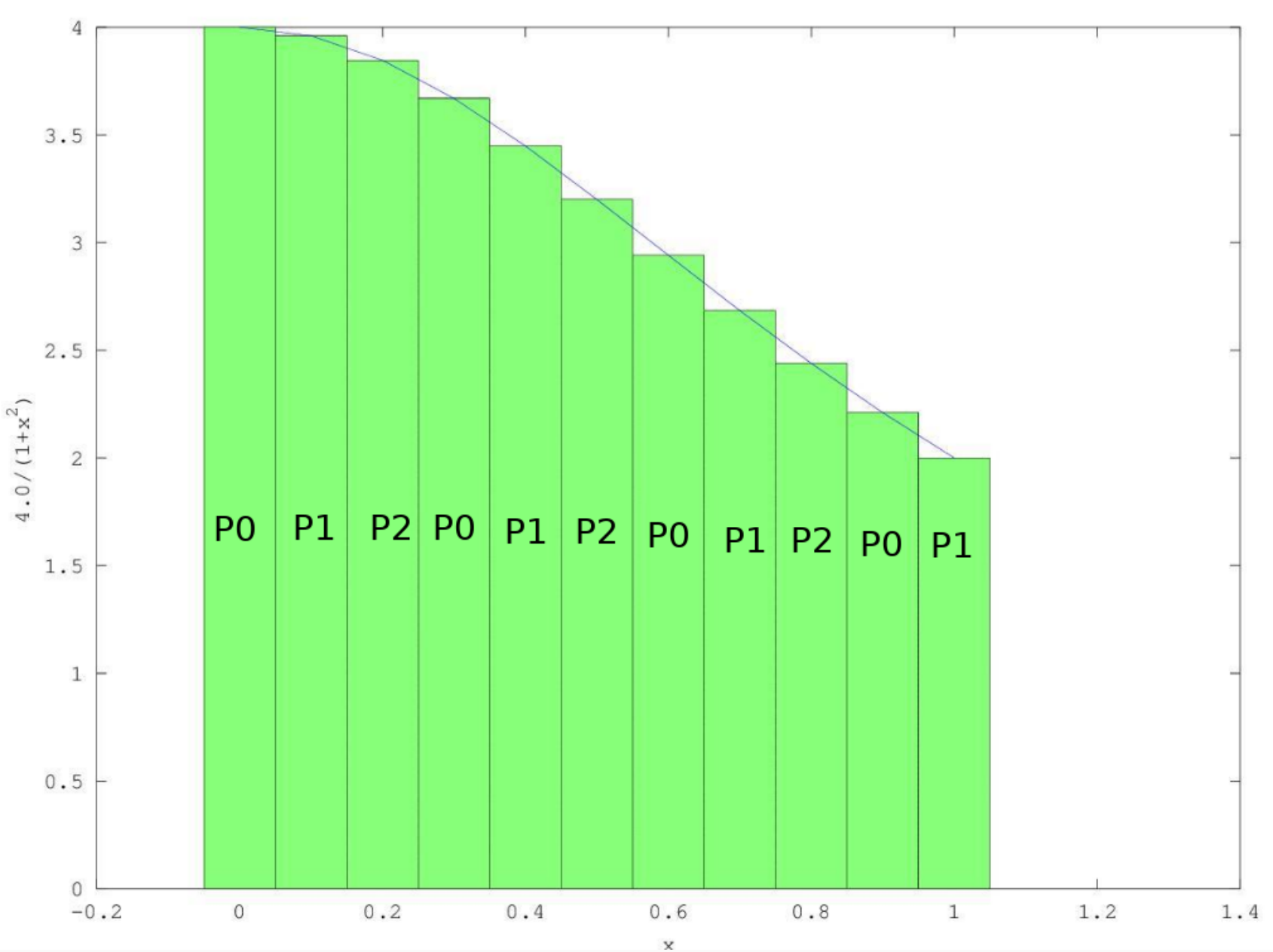
```
import time
def Pi(num_steps): # Function to calculate pi
    step = 1.0/num_steps
    sum = 0
    for i in xrange(num_steps):
        x = (i+0.5)*step
        sum = sum + 4.0/(1.0+x*x)
    pi = step * sum
    return pi

if __name__ == '__main__': # Main function
    start = time.time() # Start timing
    num_steps = 10000000
    pi=Pi(num_steps)
    end = time.time() #Stop timing
    # If this is the root process print the result
    print "Pi=%f␣(calculated␣in␣%f␣secs)" %(pi, end-start)
```

Serial Program to Calculate π



Parallel Program to Calculate π



LMod Environment Modules

HPC Systems tend to have a lot of software installed (perhaps hundreds of programs). We use two “module” environment systems to keep those programs isolated. Globally installed software is managed with LMod.

To load the MPI environment enter:

module load openmpi-3.1.1-intel-18.0.2-vde2j7x

Other useful commands:

Find modules: **module spider <search string>**
List all modules on the system: **module avail**
List loaded modules: **module list**
Unload a module: **module unload <module name>**

Anaconda Environment

The anaconda environment system allows you to create your own personalized software environment.

To create an environment called “wheeler_mpi_py2” that includes the python packages for numerical computing, scientific computing, and MPI enter:

module load anaconda to load the anaconda module

conda create --name wheeler_mpi_py2 python=2 mpi4py numpy scipy

Once the python packages have finished installing enter:

source activate wheeler_mpi_py2

Parallel Program to Calculate π

calcPiMPI.py

```
from mpi4py import MPI
import time

# Get MPI variables
comm = MPI.COMM_WORLD       # Communication framework
root = 0                    # Root process
rank = comm.Get_rank()      # Rank of this process
num_procs = comm.Get_size() # Total number of processes

# Distributed function to calculate pi
def Pi(num_steps):
    step = 1.0/num_steps
    sum = 0
    for i in xrange(rank, num_steps, num_procs):
        x = (i+0.5)*step
        sum = sum + 4.0/(1.0+x*x)
    mypi = step * sum
    pi = comm.reduce(mypi, MPI.SUM, root)
    return pi

# Main function
if __name__ == '__main__':
    start = time.time() # Start timing
    num_steps = 10000000
    # Broadcast number of steps to use to the other processes
    comm.bcast(num_steps, root);
    pi=Pi(num_steps)
    end = time.time() #Stop timing
    # If this is the root process print the result
    if (rank==root): print "Pi=%f␣(calculated␣in␣%f␣secs)" %(pi, end-start)
```

Torque Job Scheduler

Write and compile your code on the wheeler head node. Run your programs on the compute nodes.

We use the **torque** system to schedule jobs on the compute nodes.

Some useful scheduler commands:

Show current jobs: **qstat -a**
Queue information: **qstat -q** or **qgrok**
Show jobs of a particular user: **qstat -u <username>**

Submit a job: **qsub <pbs[2] script>**
Delete a job: **qdel <job ID>**

[2] portable batch system

PBS Script

calc_pi.pbs

```
#!/bin/bash

#PBS -l nodes=2:ppn=8
#PBS -l walltime=00:05:00
#PBS -N calc_pi
#PBS -S /bin/bash
#PBS -j oe
#PBS -M youremailaddress@unm.edu
#PBS -V

module load openmpi-3.1.1-intel-18.0.2-vde2j7x
module load anaconda
source activate wheeler_mpi_py2

mpirun -n $PBS_NP python calcPiMPI.py
```