High Performance Computing

Running code on giant computing clusters

COMPASS 2024

Image credit: Owen Stanley, SDSC/UC San Diego

What Is HPC?

Laptop

Supercomputer

What Is HPC?

Laptop

Supercomputer

Computing Cluster
You are here
Log-in node
Your computer
Internet
Local connections

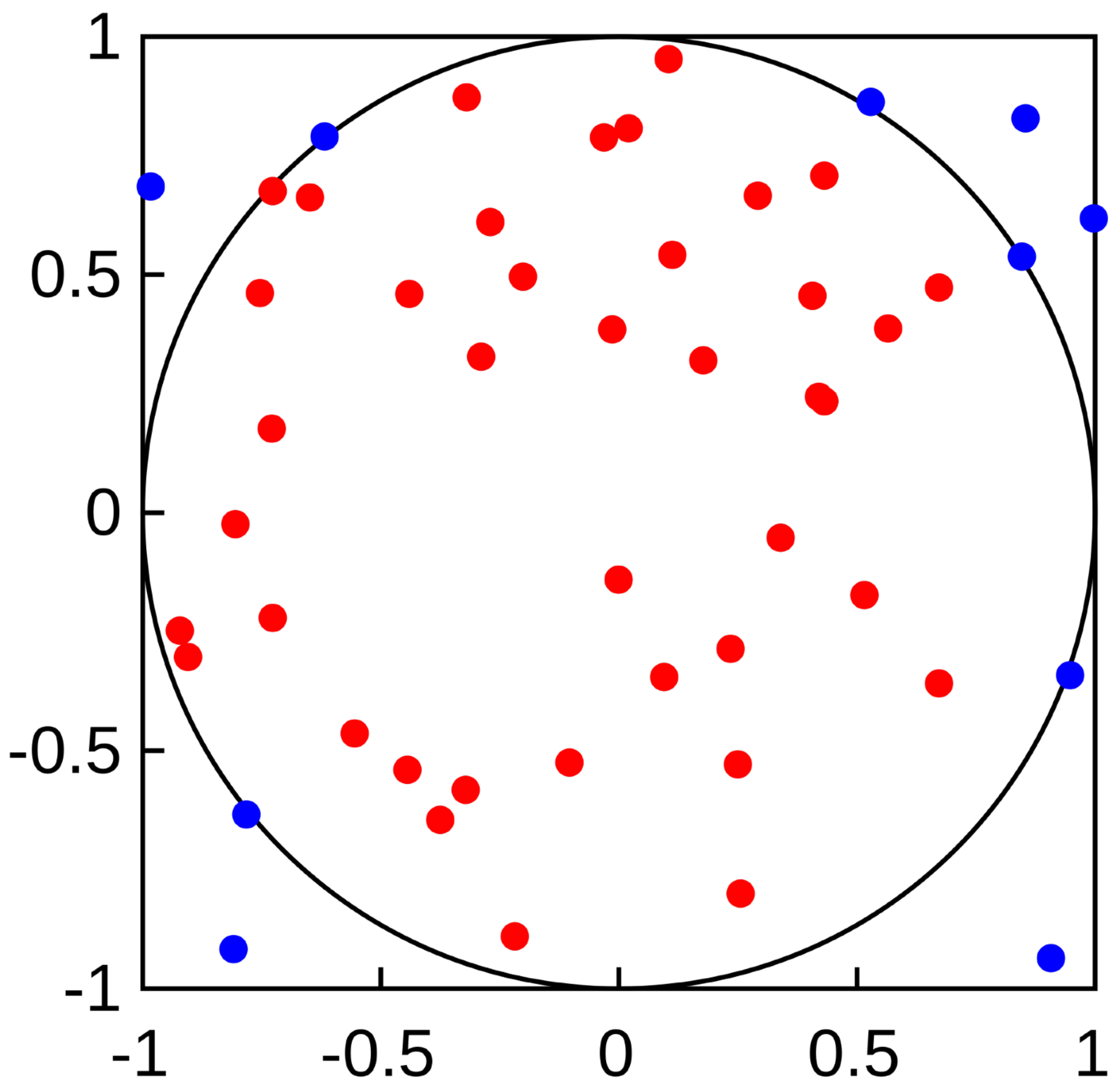
Part I: Writing some code

Part II: Running it on the HPC Cluster

Computing the value of π

(In the worst way…)

- Randomly place a dot in the square
- What is the probability of it landing inside the circle?
- Average number red dots / all dots =… ?

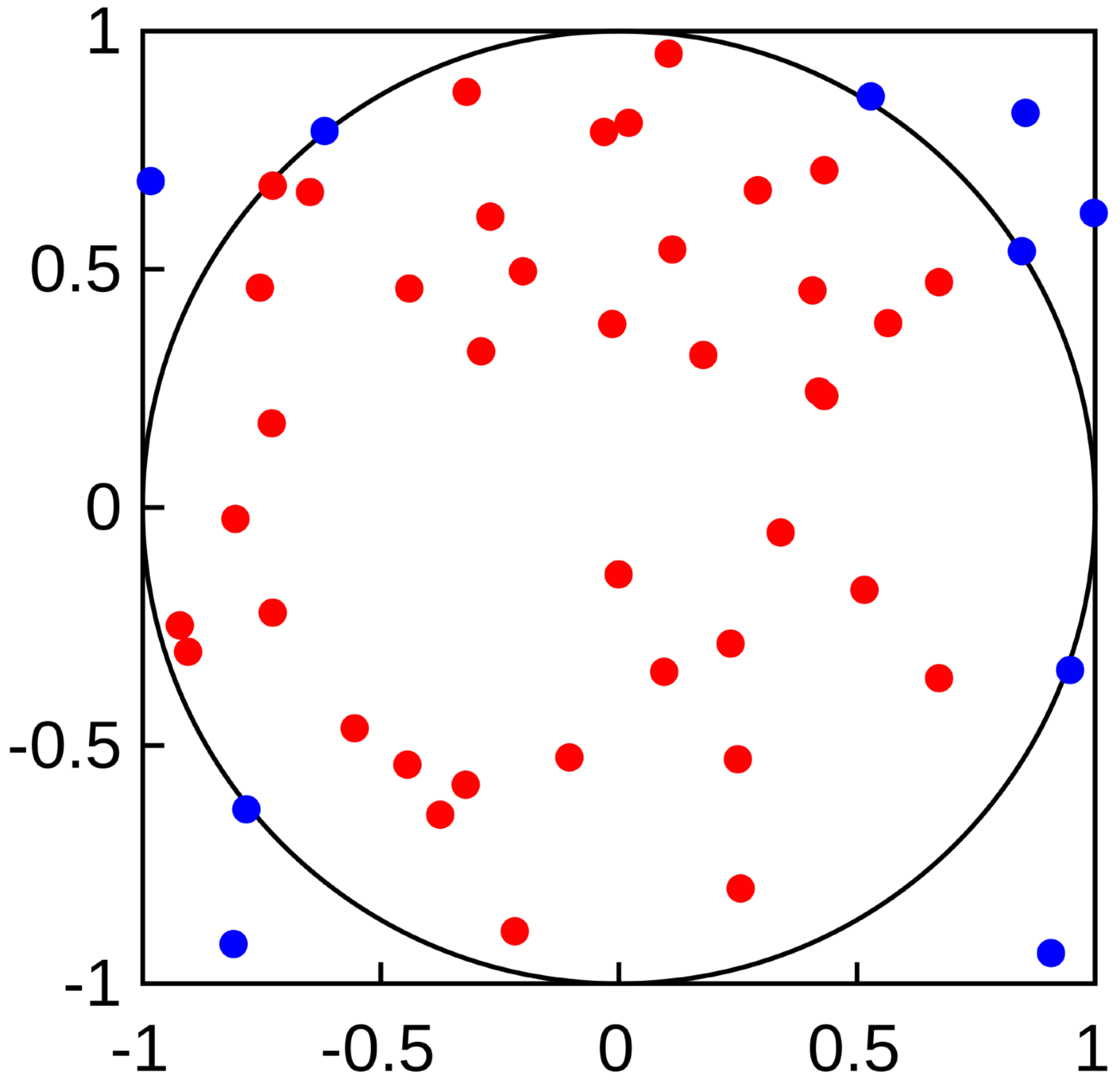


Computing the value of π

(In the worst way…)

- Randomly place a dot in the square
- What is the probability of it landing inside the circle?
- Average number red dots / all dots =… ?

$$\frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$



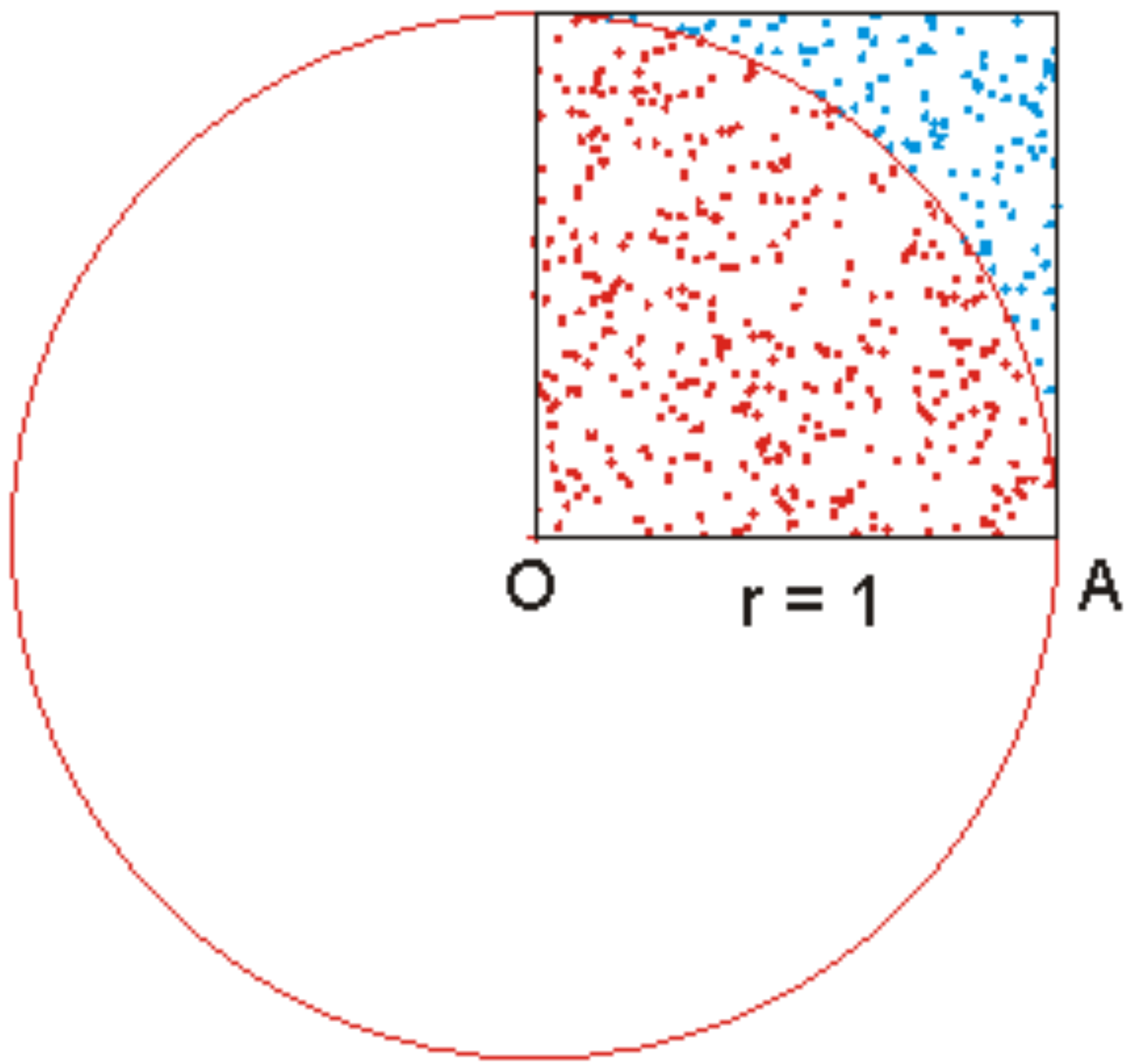
Computing the value of π

(In the worst way…)

I have written a program that:

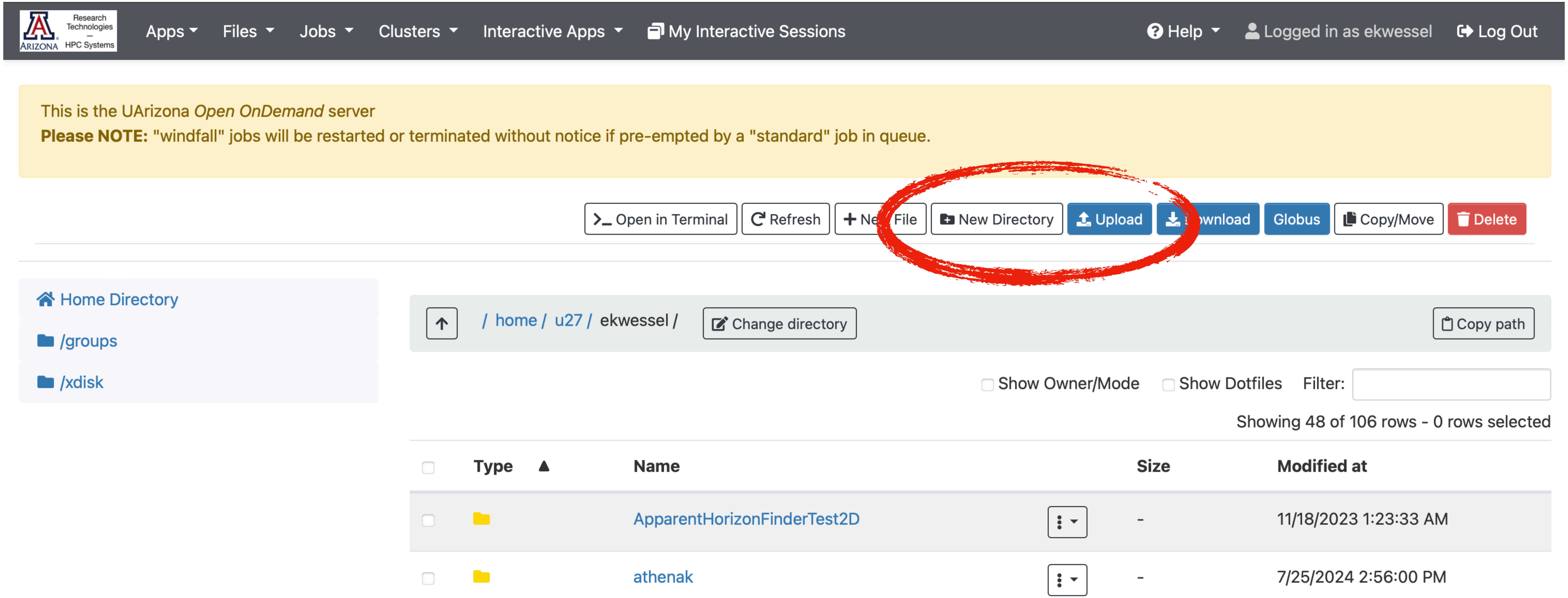
- Randomly places a points in a square
- Counts how many are within a circle
- Estimates π from the ratio
- We save effort by only doing this for one quarter of the square

This is called "Monte Carlo" integration



1. Download the script

2. Upload the `monte_carlo_pi.py` script



monte_carlo_pi.py

```
#!/usr/bin/env python3
import sys
import random

# First, we check that our script was called with the correct number of arguments
# If not, we print a usage message and exit:
if len(sys.argv) != 3:
    print("Usage: monte_carlo_pi.py n_points seed_value")
    sys.exit()

# Next, we get the two input arguments that our script was called with.
# These are the number of points to generate and the seed value for the random number generator.
n_points = int(sys.argv[1])
seed_value = int(sys.argv[2])

# Using the seed value, we initalize a random number generator
random.seed(seed_value)

# Count the number of points that fall within the unit circle
count = 0
for i in range(n_points):
    x_point = random.random()
    y_point = random.random()
    if x_point**2 + y_point**2 <= 1:
        count = count + 1

# The ratio of the area of the unit circle to the area of the unit square is pi/4
pi_estimate = 4*count/n_points

print(f"{pi_estimate:.20f}") # Print the estimate with 20 decimal places
```


You are here
Your code belongs here
Computing Cluster
Log-in node
Your computer
puma) [ekwessel@junonia ~]$
Internet
Local connections

Part I: Writing some code

Part II: Running it on the HPC Cluster

The HPC Job Scheduling Queue

- Every program users want to run is a **job**
- Users can add **jobs** to the **scheduling queue**
- The **scheduler** keeps track of the queue, decides when & where jobs run
- On UA's HPC the scheduler software is "Slurm"

Creating a job script

We use bash scripts to tell the scheduler what we want our jobs to do.

Makes this run as a bash script

Directives to the queue system

Commands to run when job starts

Based on this template, create your own job submission script: `job_sub.sh`

```
#!/bin/bash

# --------------------
### Directives Section
# --------------------
#SBATCH --job-name=hello_world
#SBATCH --account=<your_group>
#SBATCH --partition=standard
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --time=00:01:00

# --------------------
### Code Section
# --------------------
echo "Hello world, I am running on compute node $HOSTNAME"
# sleep only used for demonstration purposes
sleep 30
```

Submitting a job to the queue

- First, we must give the job script permission to execute:
  `$ chmod u+x job_sub.sh`

- To submit your job, use the **sbatch** command
  `$ sbatch job_sub.sh`

- To check on your job, use the **qstat** command
  `$ qstat -u <username>`

Submitting a bunch of jobs to the queue

- To estimate π, we need many copies of `monte_carlo_pi.py` to run on the cluster

- The best way to do this is with a **job array**

- Make a new job submission script called `job_array_sub.sh`

- Edit it, and type the following:

- `chmod u+x monte_carlo_pi.py`
  `chmod u+x job_array_sub.sh`

- Submit using sbatch:
  `$ sbatch job_array_sub.sh`

```
#!/bin/bash

#SBATCH -J monte_carlo_pi   # Job Name
#SBATCH -o monte_carlo_pi_%a.o%j
#SBATCH --partition=windfall
#SBATCH -N 1 -n 1
#SBATCH --mem-per-cpu=4gb
#SBATCH --cpus-per-task=1
#SBATCH -t 0:30:00       # Run time (hh:mm:ss)
#SBATCH --array 0-10

module load python

./monte_carlo_pi.py 1000000000 $RANDOM

```

Viewing the results

- The files `monte_carlo_pi_*.o*` contain the output
- What output did you get?
- **Exercise: How can we combine the outputs?**
- How accurate was our estimate of π?

One way to average the estimates

Save /home/u27/ekwessel/COMPASS_2024/average.py

```python
#!/usr/bin/env python3

import sys
import numpy as np

# First, we check that our script was called with the correct number of arguments
# If not, we print a usage message and exit:
if len(sys.argv) != 2:
    print("Usage: average.py filename")
    sys.exit()

filename = sys.argv[1]

estimate_array = np.genfromtxt(filename)

print(f"{np.mean(estimate_array):0.20f}")
```

UA HPC Resources

HPC Documentation:

https://hpcdocs.hpc.arizona.edu/

A wonderful introductory presentation by our HPC center staff:

https://hpcdocs.hpc.arizona.edu/events/workshop_materials/intro_to_hpc/files/IntrotoHPC.ResBaz2024.pdf