Class 12 - Monte Carlo Calculation of Ⲡ

[**Introduction to Monte Carlo Simulation:**](#_po0dkjfxsj81) **1**

[We can use Monte Carlo methods to calculate the value of Ⲡ.](#_1rsphxtdnquf) 1

[Perl Program to Calculate Π](#_l6y39t7k17pq) 3

[Python Program to Calculate Π](#_4a2tp3hsi4kj) 4

[Calculate Π HTCondor Job Template](#_fksjqava5rqv) 5

[Calculate Π CalcPi Script](#_ltzczgd91djt) 5

[Calculate Π Submit Script](#_df9ljuezwoqg) 6

[Make Π Calculation Plots](#_7bzn45xpasfx) 6

[Procedure:](#_tb51rn4mzy0c) 7

[Submit 1000 jobs:](#_70cd25ifgg22) 7

[Plotting the results:](#_vly2by9ft9ot) 8

[Congratulations:](#_ak4dh456jj2b) 10

# Introduction to Monte Carlo Simulation:

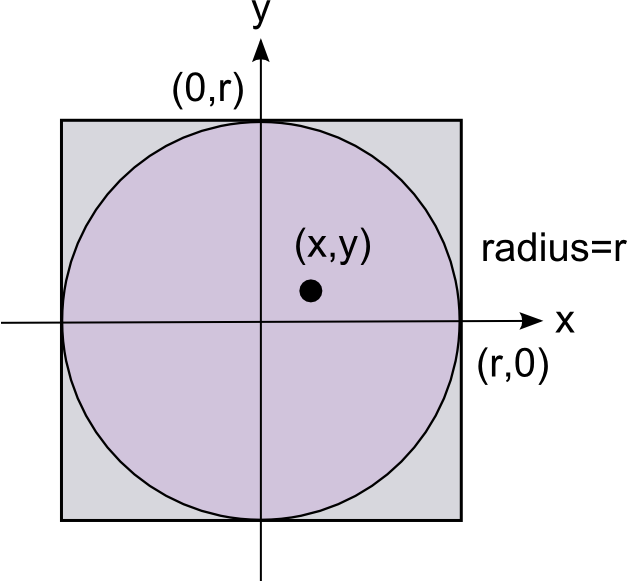
**Monte Carlo methods** **(MC)** are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. Their essential idea is using randomness to solve problems that might be deterministic in principle.

For example, MC can be use to simulate the tossing of a coin. We we assume that we toss a fair coin that has an even probability of landing on either side, we can generate a sequence of of coin flips using a Monte Carlo simulation.

|  |
| --- |
| **Question**: Assume a coin has a 50/50 chance of landing on either side. Call them Side A and Side B. If I have tossed the coin 5 times and it has landed 5 times in a row on the Side A, what is the probability that the next toss with result in Side A? |

## We can use Monte Carlo methods to calculate the value of Ⲡ.

Assume we have a square and each side has a length 2R. Now put a circle of radius R in the square, centered in the middle:



If I generate random numbers between -1 and 1 for both x and y, I can ask how many points lie in the circle versus how many points I generated. The probability of landing in the circle is the area of the circle and the probability of landing in the square is the area of the square. The number that land in the square is the total number of tries since we have constrained the choices to between -1 and 1. Therefore the ratio of the number in the circle to the total number of tries gives us an equation for Ⲡ:

Therefore:

And the error on this calculation is given by the normal distribution approximation to the binomial distribution:

Where Π is the calculated value not the known value of Π.

For fun, here is the known value of Π to 100 decimal places:

|  |
| --- |
| 3.141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825342117068 |

## Perl Program to Calculate Π

Here is a short **PERL** program that calculates Π using Monte Carlo methods:

|  |
| --- |
| #!/usr/bin/perl  use strict;  my $maxPoints = $ARGV[0] || 10;  if($maxPoints < 10) {  print " Number of points must be >= 10. Exiting...\n";  exit -1;  }  my $totalPoints;  my $pointsInCircle;  while($totalPoints < $maxPoints) {  my $x = 1.0 - rand(2);  my $y = 1.0 - rand(2);  $totalPoints++;  my $distance = ($x\*\*2 + $y\*\*2);  if($distance < 1) {  $pointsInCircle++;  }  }  my $areaSquare = 4;  my $areaCircle = ($pointsInCircle/$totalPoints)\*$areaSquare;  my $pi = $pointsInCircle/$totalPoints;  my $error = 1.96\*sqrt($pi\*(1-$pi)/$totalPoints);  $error = $error \* 4;  my $pi = $areaCircle/1\*\*2;  print "$maxPoints $pi $error\n";  exit; |

## Python Program to Calculate Π

|  |
| --- |
| **WARNING**: Indentation is part of python syntax |

Here is a short **PYTHON** program that calculates Π using Monte Carlo methods:

|  |
| --- |
| #!/usr/bin/python  import random  import math  import sys  TIMES\_TO\_REPEAT = 10  if len(sys.argv) > 1:  TIMES\_TO\_REPEAT = int(sys.argv[1])  if TIMES\_TO\_REPEAT < 10:  print " Error: Iterations must be >= 10"  sys.exit(-1)  count = inside\_count = 0.0  for i in range(TIMES\_TO\_REPEAT):  x = random.random()  y = random.random()  r = math.sqrt(math.pow(x,2.) + math.pow(y,2.))  if r < 1:  inside\_count += 1.0  count += 1.0  pi = (inside\_count / count)  error = 4.0 \* 1.96 \* math.sqrt(pi\*(1-pi)/count)  pi = 4\*pi  print " " + str(count) + " " + str(pi) + " " + str(error) |

Both programs accept one command line argument and print out:

|  |
| --- |
| Number-Of-Tries Calculated-Value-Of-Π Error-On-Calculation |

For example:

|  |
| --- |
| student> ./CalcPi.py 1000  1000.0 3.132 0.102194332336  student> ./CalcPi.pl 1000  1000 3.216 0.0984174532814175 |

Now we can submit a few thousand jobs to HTCondor and then plot the value of π and the error in the calculation versus the number of tries (points in the sample).

## Calculate Π HTCondor Job Template

Save as **CondorCalcPi.template**

|  |
| --- |
| ####################  #  # submit description file  # Example 1: queuing multiple jobs with differing  # command line arguments and output files.  #  ####################  Executable = CalcPiJob  Arguments = TRIES  Universe = vanilla  #Notification = Never  Output = CalcPiLog/CondorTestJob.$(Cluster).$(Process).out  Error = CalcPiLog/CondorTestJob.$(Cluster).$(Process).err  Log = CalcPiLog/CondorTestJob.$(Cluster).$(Process).log  ShouldTransferFiles = YES  WhenToTransferOutput = ON\_EXIT  TransferInputFiles = CalcPi  Queue |

## Calculate Π CalcPi Script

Save as **CalcPiJob**

|  |
| --- |
| #!/bin/bash  ./CalcPi $1 > CalcPi-$1.dat  exit |

Calculate Π Submit Script

Save as **SubmitCalcPi**

|  |
| --- |
| #!/usr/bin/perl  my $totalJobs = 1000;  my $maxTries = 1000000;  my $increment = int($maxTries/$totalJobs);  my $count = 0;  while($count < $totalJobs) {  $count++;  my $tries = int(10 + $count \* $increment);  system("sed s/TRIES/$tries/g CondorCalcPi.template > CondorCalcPi");  system("condor\_submit CondorCalcPi");  }  exit; |

## Make Π Calculation Plots

Save as **MakePlots.gnuplot**

|  |
| --- |
| set style line 1 lc rgb 'red' pt 5 # circle  set size 1,1  set terminal png  set output 'Calculated-Pi-vs-Tries-With-Errors.png'  set style line 1 lt 1 lw 1 pt 10 ps 0.5  set title 'Calculated Pi versus Number of Events''  set xtics 0,250000  plot "CalculatePi.dat" using 1:2:3 with errorbars ls 1  set terminal png  set output 'Calculated-Pi-vs-Tries.png'  set style line 1 lt 1 lw 1 pt 10 ps 0.5  set title 'Calculated Pi versus Number of Events''  set xtics 0,250000  plot "CalculatePi.dat" using 1:2 ls 1  set terminal png  set output 'Error-vs-Tries.png'  set style line 1 lt 1 lw 1 pt 10 ps 0.5  set title 'Error in Pi versus Number of Events''  set xtics 0,250000  plot "CalculatePi.dat" using 1:3 ls 1 |

## Procedure:

As student:

|  |
| --- |
| cd  mkdir CalculatePi  cd CalculatePi  mkdir CalcPiLog/  #  # These are extra programs you need to make and view the plots  #  sudo yum install gnuplot eog |

Save each of the files above and one of the programs to calculate Ⲡ. You should now have the files:

|  |
| --- |
| CalcPi - Program to calculate Ⲡ  CondorCalcPi.template  SubmitCalcPi  CalcPiJob  MakePlots.gnuplot |

Before you submit 1000 jobs, test CalcPi:

|  |
| --- |
| chmod +x CalcPi  ./CalcPi 500 |

You should get something like:

|  |
| --- |
| student> ./CalcPi 5000  5000 3.1512 0.0453327158950266 |

## Submit 1000 jobs:

Now you are ready to try running 1000 jobs and seeing the power of a cluster:

|  |
| --- |
| student> ./SubmitCalcPi |

With output similar to:

|  |
| --- |
| condor-wn01> ./SubmitCalcPi  Submitting job(s).  1 job(s) submitted to cluster 1153.  Submitting job(s).  1 job(s) submitted to cluster 1154.  Submitting job(s).  ... |

It will take a while for all 1000 jobs to be submitted.

|  |
| --- |
| You do not have to wait for them to finish to start looking at the results. |

## Plotting the results:

|  |
| --- |
| NOTE: If you are familiar with **Excel**, you could it to **plot** the data. On Linux, you can use **libreOffice**. |

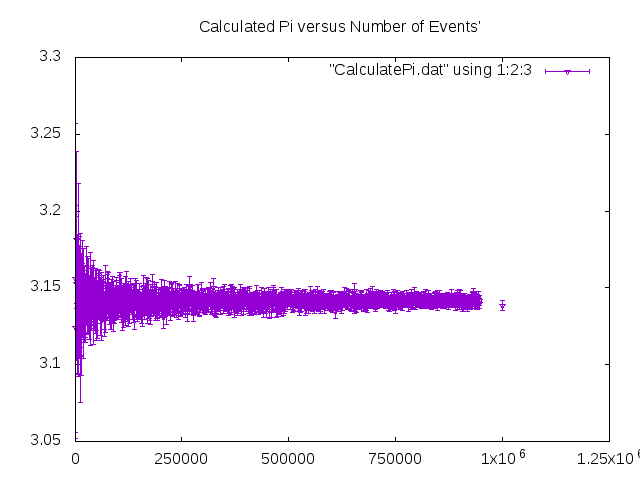
From a different window (you can repeat this procedure as the jobs are running):

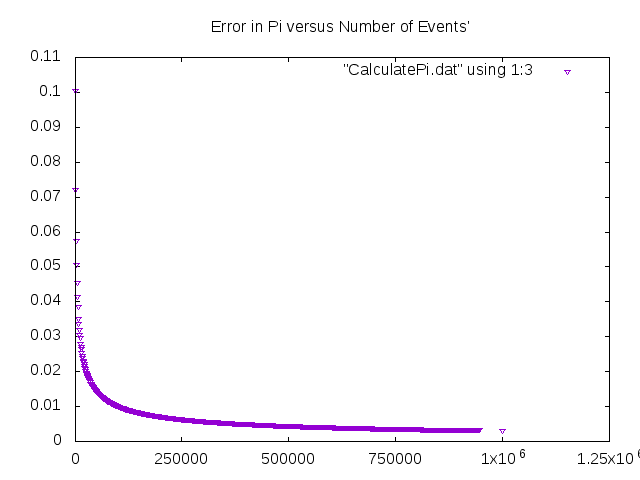
|  |
| --- |
| cat CalcPi\*dat > CalculatePi.dat  gnuplot MakePlots.gnuplot  eog Calculated-Pi-vs-Tries-With-Errors.png |

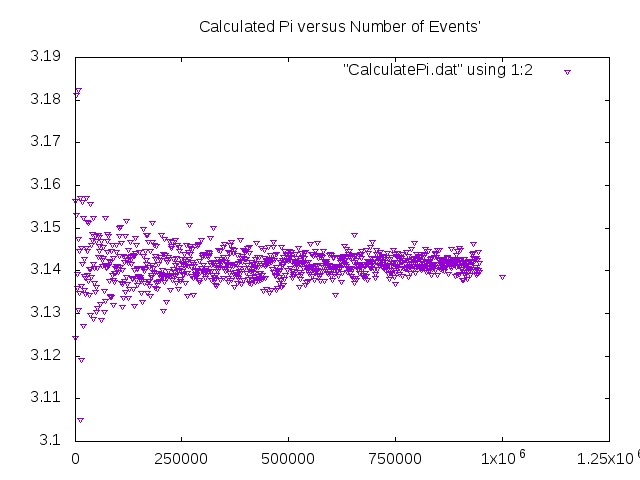
You should now have plots the three plots:

|  |
| --- |
| Calculated-Pi-vs-Tries-With-Errors.png  Calculated-Pi-vs-Tries.png  Error-vs-Tries.png |

Which should start looking like:







## Congratulations:

|  |
| --- |
| Congratulations:  If you made it this far, you have now installed, configured, managed and used a cluster computer system. |