ORIE 5270 HW6

Ko-Cheng Wang (kw582)

Repo url: https://github.coecis.cornell.edu/kw582/ORIE5270/tree/master/HW6

Files Description:

1. solution: Contain the detailed solution to HW4
2. p1a.py: Contain the solution to Problem 1, using N threads
3. p1b.py: Contain the solution to the second part of Problem 1, run in O($\sqrt{N}$) time
4. p2.py: Contain the solution to Problem 2

Problem 1)

To run the two python codes for Problem 1, use the command in the following format:

python3 p1a.py N python3 p1b.py N (e.g python3 p1a.py 1000)

For p1a.py, I utilize N threads to add the all the $\frac{1}{i^2}$ for all i from 1 to N.

For p1b.py, I first divide all N values into k groups in increasing order. Then, for each group, I utilize a thread to calculate the sum of $\frac{1}{i^2}$ in each group and then add it to the global variable. This process takes $O(k + n/k)$ time. To balance the complexity from k and n/k, I set $k = \sqrt{N}$. Hence, the total complexity would be $O(2\sqrt{N}) = O(\sqrt{N})$, which is strictly better than $O(N)$.

Problem 2)

To run p2.py, use commands in the following form:

echo 1000 | python3 p2.py

I use mrjob with two steps to compute π via monte carol simulation. The mapper function generates N random samples from uniform distribution in the range [0, 1]. Then, the reducer function checks that whether each point lies in the circle. Then, the reducer_est_pi function calculates and reports an estimate of pi based on the number of points that fell inside the circle.