HW3

November 26, 2022

Submitted by: Anubhav Shankar(01951462)

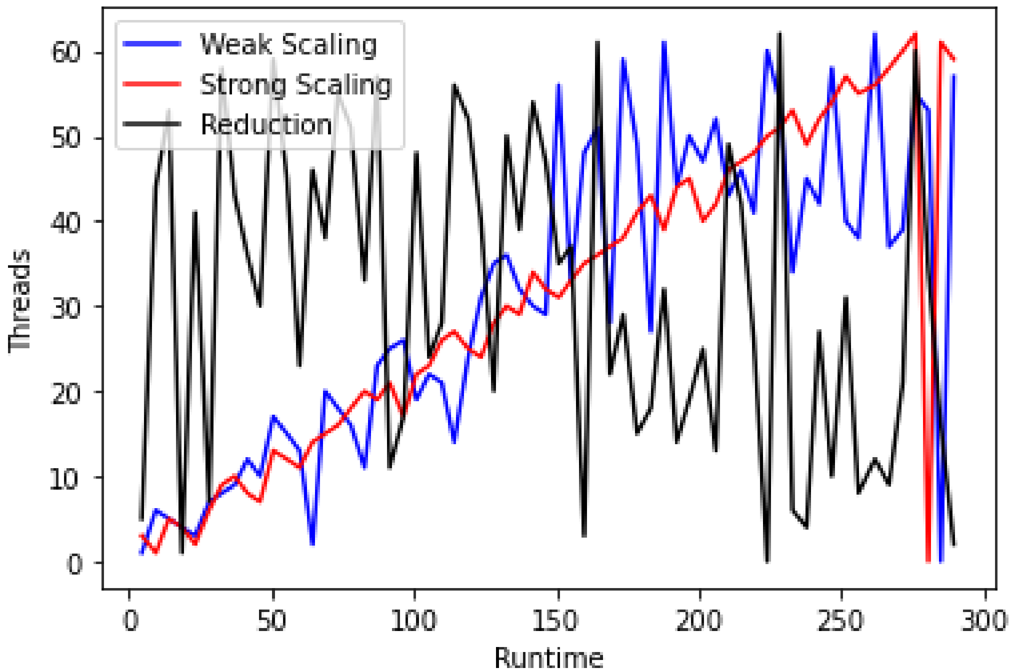
For Eqn.(4) in Problem 3, a simple test to check that the integrand has been programmed properly is to make all inputs equal to 0. The resultant value should be 'e' raised to the power negative nine.

[1]:
```
%%capture

# Load required packages
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator
import pandas as pd
import random
import sklearn as sk
from sklearn.linear_model import LinearRegression
```

[2]:
```
#Read in the .dat files

mc3_ws = np.loadtxt("mc3_omp2_critical_ws.dat")
mc3_ss = np.loadtxt("mc3_omp2_critical.dat")
mc3_red = np.loadtxt("mc3_omp2_reduction.dat")
```

[3]:
```
x = mc3_ws[:, 3]
y1 = mc3_ws[:,0]
y2 = mc3_ss[:,0]
y3 = mc3_red[:,0]
```

[4]:
```
plt.plot(x, y1,'b',label='Weak Scaling')
plt.plot(x, y2,'r',label='Strong Scaling')
plt.plot(x, y3,'black',label='Reduction')
plt.legend(loc=2)
plt.xlabel("Runtime")
plt.ylabel("Threads")
plt.savefig("montecarlo_plot.png")
```



The above graph represents three different approaches ->

1.) Critical approach with Weak Scaling

2.) Critical approach with Strong Scaling

3.) Reduction approach with Strong Scaling

From the graph, it is clear that the Reduction approach is highly volatile, and may not be suitable for our needs at the moment. However, the critical approach performs pretty well and even though the initial thread allocations seem to reduce the speedup but the trend seems to reverse with higher thread allocations. In fact the higher thread id's seem to be performing the best as evident from the steep troughs.