import os

import matplotlib.pyplot as plt

import numpy as np

if \_\_name\_\_ == "\_\_main\_\_":

all\_files = os.listdir('./')

context\_switch\_fork = [] #subarray1 for mean, 2 for variance, 3 for std dev

context\_switch\_kernel\_thread = []

measurement\_overhead = []

loop\_overhead = []

measurement\_results = []

memory\_read\_bw = [] #mean and std\_dev is Gb/s

memory\_write\_bw = []

procedure\_call\_overhead = []

ram\_time = []

ram\_overhead = []

system\_call\_overhead = []

task\_creation\_fork\_ovrhd = []

task\_creation\_Kthread\_ovrhd = []

page\_fault\_ovrhd = []

RAM = []

for file in all\_files:

if(file == "measurement\_overhead.txt"): #measurement\_overhead.txt

fp = open(file, "r")

mean = []

mean2 = []

variance = []

variance2 = []

std\_dev = []

std\_dev2 = []

for line in fp:

index\_mean = line.find("Measurement Overhead(Mean)")

if index\_mean != -1:

string = line.split(":")

mean.append(float(string[1]))

index\_variance = line.find("Measurement Overhead(Variance)")

if index\_variance != -1:

string = line.split(":")

variance.append(float(string[1]))

index\_std\_dev = line.find("Measurement Overhead(Mean)")

if index\_std\_dev != -1:

string = line.split(":")

std\_dev.append(float(string[1]))

index\_mean2 = line.find("Loop Overhead(Mean)")

if index\_mean2 != -1:

string = line.split(":")

mean2.append(float(string[1]))

index\_variance2 = line.find("Loop Overhead(Mean)")

if index\_variance2 != -1:

string = line.split(":")

variance2.append(float(string[1]))

index\_std\_dev2 = line.find("Loop Overhead(Mean)")

if index\_std\_dev2 != -1:

string = line.split(":")

std\_dev2.append(float(string[1]))

measurement\_overhead.append(mean)

measurement\_overhead.append(variance)

measurement\_overhead.append(std\_dev)

loop\_overhead.append(mean2)

loop\_overhead.append(variance2)

loop\_overhead.append(std\_dev2)

if(file == "memory\_bandwidth\_measurement.txt"): # memory bandwidth.txt

fp = open(file, "r")

mean = []

mean2 = []

variance = []

variance2 = []

std\_dev = []

std\_dev2 = []

for line in fp:

index\_mean = line.find("Memory Read Bandwidth(Mean)")

if index\_mean != -1:

string = line.split(":")

mean.append(float(string[1]))

index\_variance = line.find("Memory Read Bandwidth(Variance)")

if index\_variance != -1:

string = line.split(":")

variance.append(float(string[1]))

index\_std\_dev = line.find("Memory Read Bandwidth(Mean)")

if index\_std\_dev != -1:

string = line.split(":")

std\_dev.append(float(string[1]))

index\_mean2 = line.find("Memory Write Bandwidth(Mean)")

if index\_mean2 != -1:

string = line.split(":")

mean2.append(float(string[1]))

index\_variance2 = line.find("Memory Write Bandwidth(Mean)")

if index\_variance2 != -1:

string = line.split(":")

variance2.append(float(string[1]))

index\_std\_dev2 = line.find("Memory Write Bandwidth(Mean)")

if index\_std\_dev2 != -1:

string = line.split(":")

std\_dev2.append(float(string[1]))

memory\_read\_bw.append(mean)

memory\_read\_bw.append(variance)

memory\_read\_bw.append(std\_dev)

memory\_write\_bw.append(mean2)

memory\_write\_bw.append(variance2)

memory\_write\_bw.append(std\_dev2)

if(file == "procedure\_call\_measurement.txt"): # procedure\_call\_overhead.txt

fp = open(file, "r")

mean = []

mean2 = []

variance = []

for line in fp:

index\_mean = line.find("Procedure Call Overhead(Mean)")

if index\_mean != -1:

string = line.split(":")

mean.append(float(string[1]))

index\_variance = line.find("Procedure Call Overhead(Variance)")

if index\_variance != -1:

string = line.split(":")

variance.append(float(string[1]))

index\_std\_dev = line.find("Procedure Call Overhead(Mean)")

if index\_std\_dev != -1:

string = line.split(":")

std\_dev.append(float(string[1]))

procedure\_call\_overhead.append(mean)

procedure\_call\_overhead.append(variance)

procedure\_call\_overhead.append(std\_dev)

if(file == "ram\_access\_measurement.txt"): # ram acccess measurements.txt

fp = open(file, "r")

mean = []

mean2 = []

variance = []

for line in fp:

index\_mean = line.find("RAM Access Overhead(Mean)")

if index\_mean != -1:

string = line.split(":")

mean.append(float(string[1]))

index\_variance = line.find("RAM Access Overhead(Variance)")

if index\_variance != -1:

string = line.split(":")

variance.append(float(string[1]))

index\_std\_dev = line.find("RAM Access Overhead(Mean)")

if index\_std\_dev != -1:

string = line.split(":")

std\_dev.append(float(string[1]))

ram\_overhead.append(mean)

ram\_overhead.append(variance)

ram\_overhead.append(std\_dev)

if(file == "system\_call\_measurement.txt"): # system call measurements.txt

fp = open(file, "r")

mean = []

mean2 = []

variance = []

for line in fp:

index\_mean = line.find("System Call Overhead(Mean)")

if index\_mean != -1:

string = line.split(":")

mean.append(float(string[1]))

index\_variance = line.find("System Call Overhead(Variance)")

if index\_variance != -1:

string = line.split(":")

variance.append(float(string[1]))

index\_std\_dev = line.find("System Call Overhead(Mean)")

if index\_std\_dev != -1:

string = line.split(":")

std\_dev.append(float(string[1]))

system\_call\_overhead.append(mean)

system\_call\_overhead.append(variance)

system\_call\_overhead.append(std\_dev)

if(file == "task\_creation\_measurement.txt"): # task\_creation\_measurement.txt

fp = open(file, "r")

mean = []

mean2 = []

variance = []

variance2 = []

std\_dev = []

std\_dev2 = []

for line in fp:

index\_mean = line.find("Fork Overhead(Mean)")

if index\_mean != -1:

string = line.split(":")

mean.append(float(string[1]))

index\_variance = line.find("Fork Overhead(Variance)")

if index\_variance != -1:

string = line.split(":")

variance.append(float(string[1]))

index\_std\_dev = line.find("Fork Overhead(Mean)")

if index\_std\_dev != -1:

string = line.split(":")

std\_dev.append(float(string[1]))

index\_mean2 = line.find("Kernel Thread Overhead(Mean)")

if index\_mean2 != -1:

string = line.split(":")

mean2.append(float(string[1]))

index\_variance2 = line.find("Kernel Thread Overhead(Mean)")

if index\_variance2 != -1:

string = line.split(":")

variance2.append(float(string[1]))

index\_std\_dev2 = line.find("Kernel Thread Overhead(Mean)")

if index\_std\_dev2 != -1:

string = line.split(":")

std\_dev2.append(float(string[1]))

task\_creation\_fork\_ovrhd.append(mean)

task\_creation\_fork\_ovrhd.append(variance)

task\_creation\_fork\_ovrhd.append(std\_dev)

task\_creation\_Kthread\_ovrhd.append(mean2)

task\_creation\_Kthread\_ovrhd.append(variance2)

task\_creation\_Kthread\_ovrhd.append(std\_dev2)

if(file == "context\_switch\_measurement.txt"): # memory bandwidth.txt

fp = open(file, "r")

mean = []

mean2 = []

variance = []

variance2 = []

std\_dev = []

std\_dev2 = []

for line in fp:

index\_mean = line.find("Context Switch Fork Overhead(Mean)")

if index\_mean != -1:

string = line.split(":")

mean.append(float(string[1]))

index\_variance = line.find("Context Switch Fork Overhead(Variance)")

if index\_variance != -1:

string = line.split(":")

variance.append(float(string[1]))

index\_std\_dev = line.find("Context Switch Fork Overhead(Mean)")

if index\_std\_dev != -1:

string = line.split(":")

std\_dev.append(float(string[1]))

index\_mean2 = line.find("Context Switch Fork Overhead(Mean)")

if index\_mean2 != -1:

string = line.split(":")

mean2.append(float(string[1]))

index\_variance2 = line.find("Context Switch Fork Overhead(Mean)")

if index\_variance2 != -1:

string = line.split(":")

variance2.append(float(string[1]))

index\_std\_dev2 = line.find("Context Switch Fork Overhead(Mean)")

if index\_std\_dev2 != -1:

string = line.split(":")

std\_dev2.append(float(string[1]))

context\_switch\_fork.append(mean)

context\_switch\_fork.append(variance)

context\_switch\_fork.append(std\_dev)

context\_switch\_kernel\_thread.append(mean2)

context\_switch\_kernel\_thread.append(variance2)

context\_switch\_kernel\_thread.append(std\_dev2)

if(file == "page\_fault\_overhead.txt"): # system call measurements.txt

fp = open(file, "r")

mean = []

mean2 = []

variance = []

for line in fp:

index\_mean = line.find("Page Fault Overhead(Mean)")

if index\_mean != -1:

string = line.split(":")

mean.append(float(string[1]))

index\_variance = line.find("Page Fault Overhead(Variance)")

if index\_variance != -1:

string = line.split(":")

variance.append(float(string[1]))

index\_std\_dev = line.find("Page Fault Overhead(Mean)")

if index\_std\_dev != -1:

string = line.split(":")

std\_dev.append(float(string[1]))

page\_fault\_ovrhd.append(mean)

page\_fault\_ovrhd.append(variance)

page\_fault\_ovrhd.append(std\_dev)

# print( "measurement overhead: ", measurement\_overhead,"\n", "loop overhead: ", loop\_overhead ,'\n')

# print( "memory read overhead: ", memory\_read\_bw,"\n", "memory write overhead: ", memory\_write\_bw, '\n')

# print("proc call overhead: " , procedure\_call\_overhead, '\n')

print("ram access: ", ram\_overhead)

def normalize( input):

output = []

mx = max(input)

mn = min(input)

for i in input:

output.append((i - mn)/(mx - mn))

return output

fig, axes = plt.subplots(nrows = 3, ncols = 4)

# plt.subplots\_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)

#measurement\_overhead\_plot

axes[0,0].bar(range(len(measurement\_overhead[1])),(measurement\_overhead[1]), width=.25)

axes[0,1].bar(range(len(loop\_overhead[1])),(loop\_overhead[1]), width=.25)

axes[0,2].bar(range(len(memory\_read\_bw[1])),(memory\_read\_bw[1]), width=.25)

axes[0,3].bar(range(len(memory\_write\_bw[1])),(memory\_write\_bw[1]), width=.25)

axes[1,0].bar(range(len(procedure\_call\_overhead[0])),(procedure\_call\_overhead[0]), width=.25)

axes[1,1].bar(range(len(ram\_overhead[0])),(ram\_overhead[0]), width=.25)

axes[1,2].bar(range(len(system\_call\_overhead[0])),(system\_call\_overhead[0]), width=.25)

axes[1,3].bar(range(len(task\_creation\_fork\_ovrhd[0])),(task\_creation\_fork\_ovrhd[0]), width=.25)

axes[2,0].bar(range(len(task\_creation\_Kthread\_ovrhd[0])),(task\_creation\_Kthread\_ovrhd[0]), width=.25)

axes[2,1].bar(range(len(context\_switch\_fork[0])),(context\_switch\_fork[0]), width=.25)

axes[2,2].bar(range(len(context\_switch\_kernel\_thread[0])),(context\_switch\_kernel\_thread[0]), width=.25)

axes[2, 3].bar(range(len(page\_fault\_ovrhd[0])),(page\_fault\_ovrhd[0]), width=.25)

axes[0,0].set\_title('measurement overhead')

axes[0,1].set\_title('loop overhead')

axes[0,2].set\_title('memory read bandwidth')

axes[0,3].set\_title('memory write bandwidth')

axes[1,0].set\_title('procedure call overhead')

axes[1,1].set\_title('RAM access overhead')

axes[1,2].set\_title('system call overhead')

axes[1,3].set\_title('task creation fork overhead')

axes[2,0].set\_title('task creation Kthread overhead')

axes[2,1].set\_title('contect switch fork overhead')

axes[2,2].set\_title('context switch kthread overhead')

axes[2, 3].set\_title('page fault overhead')

plt.subplots\_adjust(left=0.05, bottom=0.04, right=0.98,

top=0.94, wspace=0.21, hspace=0.22)

plt.show()

# fp = open(all\_files[0], "r")

# storage = []

# for line in fp:

# index = line.find("Context Switch Fork Overhead(Mean)")

# if index != -1:

# string = line.split(":")

# storage.append(float(string[1]))

# plt.bar(range(len(storage)), storage, width=.25)

# plt.show()

# print(storage)