SEGUNDO PROYECTO

COMPARATIVA DE ALGORITMOS SERIALES Y PARALELOS

## Alumno: Robles Martínez Héctor

## Número de Cuenta UNAM: 317356548

## Semestre: Tercer semestre de Ingeniería en Computación (2016)

## Semestre Actual: 2021-1

## Profesor: Jorge A. Solano G.

Objetivo:

Crear una aplicación que permita realizar procesamiento paralelo para instancias de entrada grandes (miles de datos).

Descripción:

El proyecto consiste en proponer e implementar una aplicación que se pueda paralelizar.

La implementación se debe realizar en Java y en lenguaje C con OpenMP, tanto de forma serial como de forma paralela, es decir, el algoritmo se debe implementar en:

1. Java de forma serial.

2. Java con hilos de forma paralela.

3. Lenguaje C de forma serial.

4. Lenguaje C y OpenMP de forma paralela.

Para cada implementación realizada se debe contabilizar el tiempo de ejecución para diferentes instancias de datos ([1, 10,000]). Con los tiempos contabilizados en las diferentes implementaciones se debe realizar la grafica de las curvas que tiene el algoritmo y hacer una comparativa del comportamiento de las implementaciones para hallar el punto de equilibrio. Las gráficas se pueden realizar en cualquier lenguaje de programación (se recomienda Python).

La aplicación no puede ser ninguna de las vistas en clase. Las aplicaciones no se pueden parecer entre las propuestas del grupo (cada quien tiene que realizar una implementación única).

Para este segundo proyecto, planteo el realizar el cálculo del número e mediante su forma de series, la cual se define como:

Donde voy a paralelizar el cálculo del factorial de n y la serie, esto lo haré utilizando vectores donde pueda almacenar diferentes secciones de estos cálculos, todos en rangos distintos para después juntarlos todos, que sería la técnica de “Divide y vencerás”, mientras que para la parte serial, usaré el método usual para calcular un factorial, que consiste en multiplicar todos los números antes de este.

Cabe destacar que el cáclulo se realiza para obtener un número más exacto, debido a que desde el tercer cálculo, donde n = 3, se obtiene un número parecido a e, sin embargo, sólo coinciden los primeros decimales.

|  |  |  |
| --- | --- | --- |
| n | C | Java |
| 0 | 1 | 1 |
| 1 | 2 | 2 |
| 2 | 2.5 | 2.5 |

Código python para graficar los tiempos de procesamiento:

import pandas as pd

import matplotlib.pyplot as plt

import matplotlib.patches as patches

import random

def makeLists(data):

x1 = []

x2 = []

for i in range(0, len(data), 2):

x1.append(data[i])

x2.append(data[i + 1])

return x1, x2

def makeDeltas(data1, data2): #serial, paralelo

len1 = len(data1)

len2 = len(data2)

lenAbs = len1 if len1 < len2 else len2

delta1 = [] #serial

delta2 = [] #paralelo

delta3 = [] #igual

deltaX = []

for i in range(0, lenAbs):

delta = data1[i] - data2[i]

if 0.001 <= delta and delta <= 0.002:

delta1.append(None)

delta2.append(data1[i])

delta3.append(None)

elif -0.002 <= delta and delta <= -0.001:

delta1.append(data2[i])

delta2.append(None)

delta3.append(None)

elif -0.001 < delta and delta < 0.001:

delta1.append(None)

delta2.append(None)

delta3.append(delta)

else:

delta1.append(None)

delta2.append(None)

delta3.append(None)

deltaX.append(i)

return delta1, delta2, delta3, deltaX

def main():

delta1 = []

delta2 = []

delta3 = []

deltaX = []

xC = []

xCSerial = []

xCParalelo = []

yC = []

yCSerial = []

yCParalelo = []

xJava = []

xJavaSerial = []

xJavaParalelo = []

yJava = []

yJavaSerial = []

yJavaParalelo = []

dataC = pd.read\_csv("../dataC.csv", names=('e', 'y', 'x', 'NA'))

dataJava = pd.read\_csv("../dataJava.csv", names=('e', 'y', 'x', 'NA'))

xC = dataC['x'].tolist()

xCSerial, xCParalelo = makeLists(xC)

yC = dataC['y'].tolist()

yCSerial, yCParalelo = makeLists(yC)

xJava = dataJava['x'].tolist()

xJavaSerial, xJavaParalelo = makeLists(xJava)

yJava = dataJava['y'].tolist()

yJavaSerial, yJavaParalelo = makeLists(yJava)

fig = plt.figure(figsize=(30, 25))

ax1 = fig.add\_subplot(3, 3, 1)

ax1.set\_ylabel("C")

ax1.set\_xlabel("Paralelo")

ax1.plot(xCParalelo, yCParalelo, "b", label="C paralelo")

ax1.legend(loc="best")

ax2 = fig.add\_subplot(3, 3, 2)

ax2.plot(xCParalelo, yCParalelo, "b", label="C paralelo")

ax2.plot(xCSerial, yCSerial, "g", label="C serial")

delta1, delta2, delta3, deltaX = makeDeltas(yCParalelo, yCSerial)

ax2.plot(deltaX, delta1, "c.", label="Es mejor paralelo")

ax2.plot(deltaX, delta2, "m.", label="Es mejor serial")

ax2.plot(deltaX, delta3, "y.", label="Son casi iguales")

delta1 = [x if x is not None else 0.0015 for x in delta1]

delta2 = [x if x is not None else -0.0015 for x in delta2]

delta3 = [x if x is not None else 0 for x in delta3]

rect1 = patches.Rectangle((delta1.index(min(delta1)), 0), (deltaX[-1]-delta1.index(min(delta1))), 0.5, color='c', alpha=0.3)

rect2 = patches.Rectangle((delta2.index(min(delta2)), 0), deltaX[delta2.index(max(delta2))], 0.5, color='m', alpha=0.3)

rect3 = patches.Rectangle((delta3.index(min(delta3)), 0), deltaX[delta3.index(max(delta3))], 0.5, color='y', alpha=0.3)

ax2.add\_patch(rect1)

ax2.add\_patch(rect2)

ax2.add\_patch(rect3)

ax2.legend(loc="best")

ax3 = fig.add\_subplot(3, 3, 3)

ax3.set\_xlabel("Serial")

ax3.plot(xCSerial, yCSerial, "g", label="C serial")

ax3.legend(loc="best")

ax4 = fig.add\_subplot(3, 3, 4)

ax4.set\_ylabel("C vs Java")

ax4.set\_xlabel("Paralelo")

ax4.plot(xCParalelo, yCParalelo, "b", label="C paralelo")

ax4.plot(xJavaParalelo, yJavaParalelo, "m", label="Java paralelo")

delta1, delta2, delta3, deltaX = makeDeltas(yCParalelo, yJavaParalelo)

ax4.plot(deltaX, delta1, "c.", label="Es mejor Java")

ax4.plot(deltaX, delta2, "m.", label="Es mejor C")

ax4.plot(deltaX, delta3, "y.", label="Son casi iguales")

delta1 = [x if x is not None else 0.0015 for x in delta1]

delta2 = [x if x is not None else -0.0015 for x in delta2]

delta3 = [x if x is not None else 0 for x in delta3]

rect1 = patches.Rectangle((delta1.index(min(delta1)), 0), (deltaX[-1]-delta1.index(min(delta1))), 0.5, color='c', alpha=0.3)

rect2 = patches.Rectangle((delta2.index(min(delta2)), 0), deltaX[delta2.index(max(delta2))], 0.5, color='m', alpha=0.3)

rect3 = patches.Rectangle((delta3.index(min(delta3)), 0), deltaX[delta3.index(max(delta3))], 0.5, color='y', alpha=0.3)

ax4.add\_patch(rect1)

ax4.add\_patch(rect2)

ax4.add\_patch(rect3)

ax4.legend(loc="best")

ax5 = fig.add\_subplot(3, 3, 5)

ax5.set\_xlabel("Comparativa")

ax5.plot(xCParalelo, yCParalelo, "b", label="C paralelo")

ax5.plot(xCSerial, yCSerial, "g", label="C serial")

ax5.plot(xJavaParalelo, yJavaParalelo, "m", label="Java paralelo")

ax5.plot(xJavaSerial, yJavaSerial, "y", label="Java serial")

ax5.legend(loc="best")

ax6 = fig.add\_subplot(3, 3, 6)

ax6.set\_xlabel("Serial")

ax6.plot(xCSerial, yCSerial, "g", label="C Serial")

ax6.plot(xJavaSerial, yJavaSerial, "y", label="Java serial")

delta1, delta2, delta3, deltaX = makeDeltas(yCSerial, yJavaSerial)

ax6.plot(deltaX, delta1, "c.", label="Es mejor Java")

ax6.plot(deltaX, delta2, "m.", label="Es mejor C")

ax6.plot(deltaX, delta3, "y.", label="Son casi iguales")

delta1 = [x if x is not None else 0.0015 for x in delta1]

delta2 = [x if x is not None else -0.0015 for x in delta2]

delta3 = [x if x is not None else 0 for x in delta3]

rect1 = patches.Rectangle((delta1.index(min(delta1)), 0), (deltaX[-1]-delta1.index(min(delta1))), 0.5, color='c', alpha=0.3)

rect2 = patches.Rectangle((delta2.index(min(delta2)), 0), deltaX[delta2.index(max(delta2))], 0.5, color='m', alpha=0.3)

rect3 = patches.Rectangle((delta3.index(min(delta3)), 0), deltaX[delta3.index(max(delta3))], 0.5, color='y', alpha=0.3)

ax6.add\_patch(rect1)

ax6.add\_patch(rect2)

ax6.add\_patch(rect3)

ax6.legend(loc="best")

ax7 = fig.add\_subplot(3, 3, 7)

ax7.set\_ylabel("Java")

ax7.set\_xlabel("Paralelo")

ax7.plot(xJavaParalelo, yJavaParalelo, "m", label="Java paralelo")

ax7.legend(loc="best")

ax8 = fig.add\_subplot(3, 3, 8)

ax8.set\_xlabel("Comparativa")

ax8.plot(xJavaParalelo, yJavaParalelo, "m", label="Java paralelo")

ax8.plot(xJavaSerial, yJavaSerial, "y", label="Java serial")

delta1, delta2, delta3, deltaX = makeDeltas(yJavaSerial, yJavaParalelo)

ax8.plot(deltaX, delta1, "c.", label="Es mejor paralelo")

ax8.plot(deltaX, delta2, "m.", label="Es mejor serial")

ax8.plot(deltaX, delta3, "y.", label="Son casi iguales")

delta1 = [x if x is not None else 0.0015 for x in delta1]

delta2 = [x if x is not None else -0.0015 for x in delta2]

delta3 = [x if x is not None else 0 for x in delta3]

rect1 = patches.Rectangle((delta1.index(min(delta1)), 0), (deltaX[-1]-delta1.index(min(delta1))), 0.5, color='c', alpha=0.3)

rect2 = patches.Rectangle((delta2.index(min(delta2)), 0), deltaX[delta2.index(max(delta2))], 0.5, color='m', alpha=0.3)

rect3 = patches.Rectangle((delta3.index(min(delta3)), 0), deltaX[delta3.index(max(delta3))], 0.5, color='y', alpha=0.3)

ax8.add\_patch(rect1)

ax8.add\_patch(rect2)

ax8.add\_patch(rect3)

ax8.legend(loc="best")

ax9 = fig.add\_subplot(3, 3, 9)

ax9.set\_xlabel("Serial")

ax9.plot(xJavaSerial, yJavaSerial, "y", label="Java serial")

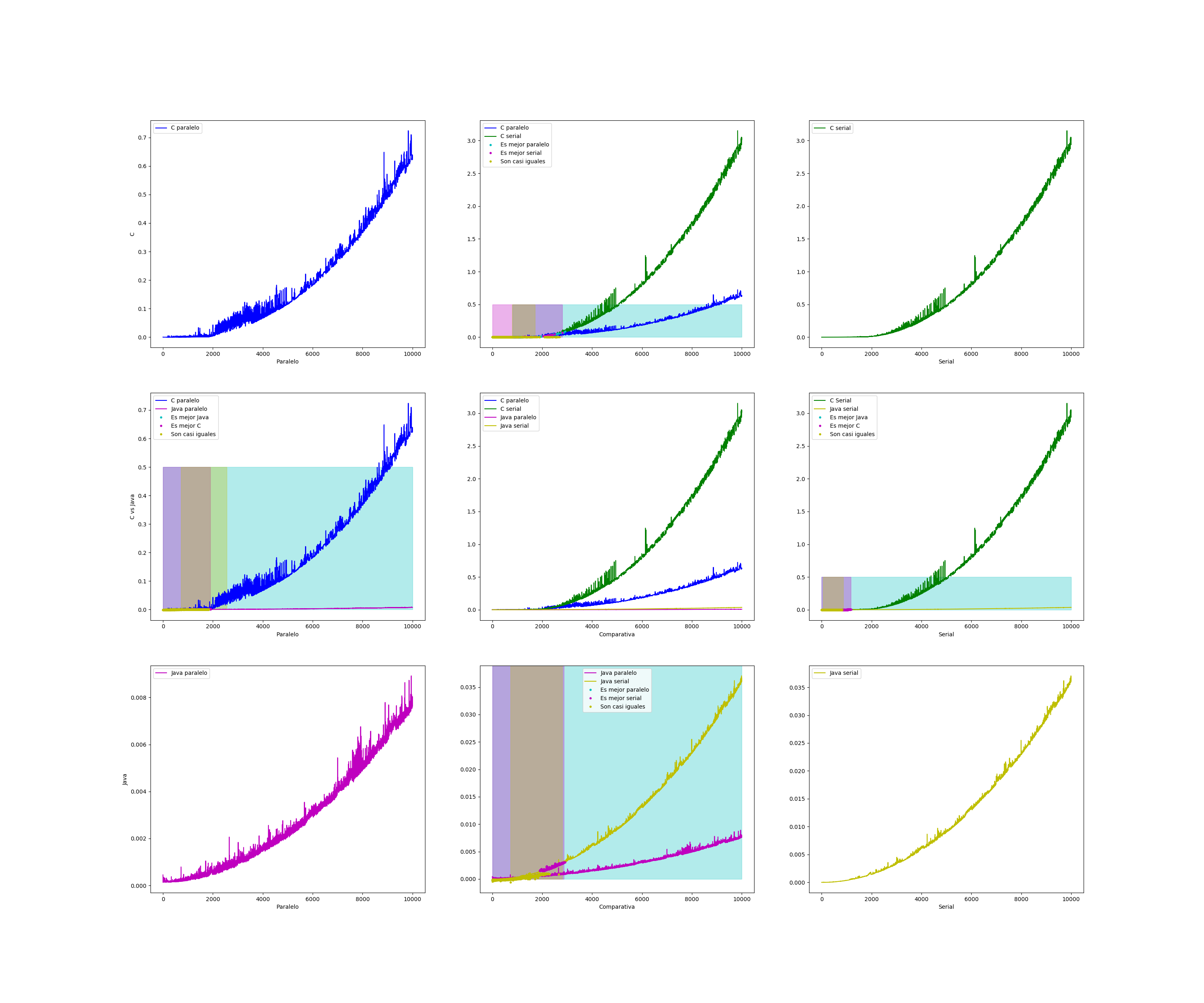
ax9.legend(loc="best")

plt.savefig("../comp.png")

plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

main()



Conclusiones:

En este segundo proyecto, pude observar por qué no siempre conviene paralelizar un proyecto, debe tenerse cuidado para no hacerlo más lento, al igual que puse observar que es importante escoger bien el lenguaje en el que se hará un proyecto.