Aplicando Paralelismo – Método del trapecio para cálculo de áreas bajo la curva

Repositorio: https://github.com/charliemike3124/IntegracionEnParalelo

Grupo: Carlos Villalobos – Omar Zuluaga

Cuellos de Botella

El cuello de botella más evidente en este ejercicio se produce cuando se quiere imprimir el área de cada trapecio en la terminal.

```
print('Hilo: ', str(comm.rank), "Trapecio: " , i, "Area: " , integral(data['a'], data['b'], i))
```

El cálculo del área del trapecio como tal no es la parte que más limita al programa, cómo se pensaría normalmente, si no la impresión de todo el texto en pantalla.

Variables Privadas

- start_time y total_time: Variables utilizadas para medir el tiempo de ejecución de cada hilo individual.

Variables Compartidas

- a y b: límites inferior y superior.
- Tramos: Número de trapecios que se quieren procesar en la integral.
- Trabajos: Cantidad de trabajos (tasks) que se van a hacer entre todos los hilos(igual al número de tramos +1).
- i: contador del ciclo para llevar cuenta de cual trabajo se debe hacer.

Bloques No Paralelizables

El único bloque no paralelizable en este ejercicio específico es la definición de la función que suma las integrales.

```
functionx = lambda x : np.cos(x) + x**3

def integral(a, b, tramos):
    h = (b - a) / tramos
    x = a
    suma = functionx(x)
    for i in range(0, tramos - 1, 1):
        x = x + h
        suma = suma + 2 * functionx(x)
        suma = suma + functionx(b)
        area = h * (suma / 2)
        return area
```

Hotspots

- Ejecución de la función de integral en paralelo:

```
for i,task in enumerate(data['trabajos']):
    if i%size!=rank: continue
    for j in range(1, int(data['tramos'])):
        if i!= 0:
            print('Hilo: ', str(comm.rank), "Trapecio: " , i, "Area: " , integral(data['a'], data['b'], i))
        break
```

Aquí es donde más se trabaja ya que todos los hilos están ejecutando la función de la integral.

Tareas

El número de tareas se define en la variable “trabajos”. La cantidad de trabajos depende del número de trapecios que define el usuario. Estas tareas luego se dividen entre cada hilo de la siguiente manera:

```
trabajos = range(int(tramos)+1)

for i,task in enumerate(data['trabajos']):
    if i%size!=rank: continue
```

Tamaño de las Tareas

La cantidad de tareas se definen por la cantidad de trapecios que se van a calcular. También, el número del trapecio puede representar un mayor cálculo del área, aunque esta diferencia no es muy notoria.

Necesidad de Comunicación

La comunicación es necesaria para este ejercicio. Sin tener comunicación entre el hilo principal y los demás sería imposible tener las variables necesarias para la ejecución de la integral.

```
if comm.rank==0:
    a = 1
    b = 10
    tramos = input("Numero de Trapecios: ")
    trabajos = range(int(tramos)+1)
    data = {'a':a,'b':b,'tramos':tramos,'trabajos':trabajos}
else:
    data = None

start_time = time.process_time() #el tiempo se mide después de

data = comm.bcast(data,root=0)
```

En el código se crean las variables únicamente en el rango 0 (el rango principal), y luego se transmite por medio de un “Broadcast” a los demás procesos.

Sincronización

La función para calcular la integral funciona de manera asíncrona. El único bloque donde hay sincronización de hilos es para sumar los tiempos de ejecución una vez la integral ya ha terminado.

```
total_time = time.process_time() - start_time
run_time = comm.gather(total_time,0)
if comm.rank == 0:
    print(run_time)
    print(np.sum(run_time))
```

La sincronización se hace con la función “gather”. El proceso 0 toma los valores del tiempo de ejecución de todos los hilos y luego los suma y los imprime en pantalla.

Tipo de Comunicación

La comunicación en este programa es asíncrona, ya que los hilos saben qué tarea hacer sin necesidad de que los otros hilos se lo comuniquen directamente. Este proceso se logra con la siguiente línea de código:

```
for i,task in enumerate(data['trabajos']):
    if i%size!=rank: continue
    for j in range(1, int(data['tramos'])):
        if i!= 0:
            print('Hilo: ', str(comm.rank), "Trapecio: " , i, "Area: " , integral(data['a'], data['b'], i))
        break
```

Comportamiento con n = 1, n = 10, n = 1,000 y n = 1, 000,000.

La toma de datos se hizo con 8 procesos en paralelo.

	Tiempo en Serial	Tiempo en Paralelo
N = 10	0.00023 s	0.00024 s
N = 100	0.0014 s	0.00064 s
N = 1000	0.0224 s	0.00065 s
N = 1000000	38 segundos	25 segundos

El tiempo de ejecución cuando la carga para el servidor es baja (n = 10, 100,1000) no es tan diferente en ambos el serial y el paralelo. Pero cuando la carga es alta podemos ver una diferencia notoria de más de 10 segundos en el tiempo de ejecución.

Hay que tener en cuenta que los tiempos de ejecución en el servidor a veces varían mucho dependiendo de si alguien más está recargando el servidor con otros procesos ajenos.