Evaluación de una función de manera paralela

Valeria Jahzeel Castañón Hernández
Ingenieria en Inteligencia Artificial - Cómputo Paralelo
Instituto Politécnico Nacional
chj068378@gmail.com

1 Descripción del problema

Se cuenta con una lista de n elementos que se quiere evaluar en una función lambda cualquiera, por lo que se requiere de un programa que permita evaluar los valores de la lista en dicha función pero de manera paralela.

2 Marco teórico

2.1 Programación paralela

La programación paralela es un paradigma de programación en el que múltiples tareas se ejecutan simultáneamente. En lugar de ejecutar instrucciones de forma secuencial, como en la programación convencional, en la programación paralela se aprovechan los recursos de hardware disponibles para realizar múltiples cálculos al mismo tiempo.

Este enfoque es especialmente útil en sistemas con múltiples núcleos de procesamiento, ya que permite distribuir la carga de trabajo entre esos núcleos, mejorando significativamente el rendimiento y la eficiencia del sistema. La programación paralela se utiliza en una variedad de campos, incluyendo la informática de alto rendimiento, la simulación científica, el procesamiento de datos a gran escala y la computación en la nube, entre otros.

2.2 Nivelación de cargas

La nivelación de cargas es un concepto fundamental en la programación paralela que se refiere a la distribución equitativa de la carga de trabajo entre los distintos procesadores o núcleos de un sistema paralelo. El objetivo de la nivelación de cargas es maximizar la utilización de los recursos disponibles y minimizar el tiempo total de ejecución de una tarea.

En entornos paralelos, es común que algunas partes del programa requieran más tiempo de procesamiento que otras, lo que puede generar desequilibrios en la distribución de la carga entre los diferentes procesadores. Esto puede llevar a una utilización ineficiente de los recursos y a tiempos de ejecución prolongados.

2.3 Funciones lambda

Las funciones lambda, también conocidas como funciones anónimas o funciones de flecha, son una característica importante en la programación funcional y en algunos lenguajes de programación orientados a objetos. Una función lambda es una función sin nombre que puede ser definida en línea y utilizada en el lugar donde se necesita, sin la necesidad de asignarle un nombre explícito.

Las funciones lambda son particularmente útiles cuando se necesitan funciones simples y de corta duración que se utilizarán como argumentos de otras funciones o que se necesitan localmente en un bloque de código. A continuación se presenta un ejemplo:

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

$$\lambda(x, y) \rightarrow x + y$$

La función lambda suma dos números: $(\lambda(x, y) \rightarrow x + y)(3, 5) = 8$.

3 Solución propuesta al problema

1. **Declaración de variables:** Inicializar las variables al inicio del programa para una fácil modificación en caso de que se requiera.

```
lim_inf = 0
lim_sup = 10
cant_valores = 9
num_procesadores = 2

```

2. **Declaración de las funciones:** Se declaran las funciones que se utilizarán en el código.
   - ***Función para la nivelación de cargas:*** Se utilizó una función basada en lo que se vio en clase sobre proponer una solución al problema de repartir datos de la manera mas equitativa entre el número de procesadores que se tienen disponibles.

```
def nivelacion_cargas(valores, procesadores):
    mod = len(valores) % procesadores
    ind = math.floor(len(valores)/procesadores )
    final = []
    li = 0
    ls = ind
    for i in range(procesadores):
        ls = li+ind
        if i < mod:    #
            ls += 1
        final.append(valores[li:ls])
        li = ls
    return final

```

   - ***Función para la generación de números aleatorios en un rango de valores:*** Se utilizó el generador por defecto de la libreria numpy, en donde se da el limite superior, inferior y la cantidad de valores que se quieren generar. El hecho de que el arreglo sea de tipo numpy se verá mas adelante.

```
def aleatorios_rango(inf,sup,cant_valores):
    lista = np.random.randint(inf,sup,cant_valores)
    return lista

```

3. **Programa principal:** Ya declaradas las funciones solo queda mandarlas a llamar en el prgorama principal. Se utiliza la funciónPool.map de la libreria multiprocessing para realizar las programación en paralelo, es aquí en donde tiene sentido el haber creado el arreglo de números aleatorios de tipo numpy desde el inicio ya que las funciones lambda no aceptaban otro tipo de datos, si se colocaba un tipo de dato diferente se requeria hacer otros procesos. Al último se hace una conversión de la lista con los resultados a un arreglo de una dimensión.

```
lista = aleatorios_rango(lim_inf, lim_sup, cant_valores)

cargas = nivelacion_cargas(lista, num_procesadores)

funcion = lambda x: x ** 2

res = []
for i in range(len(cargas)):
     with Pool(num_procesadores) as pool:
          res.append(pool.map(funcion, cargas[i]))

```

```
            resultado = [item for mini_lista in res for item in
        mini_lista]
```

4 Resultados

Suponiendo que queremos que nuestra lista contenga 10 números aleatorios entre 0 y 100. Y que además contamos con 5 procesadores, tenemos:

```
Lista -->  [34 39 46  7 62 64 66 84 54 64]
Resultado -->  [1156, 1521, 2116, 49, 3844, 4096, 4356, 7056, 2916, 4096]
```

En caso de que se requieran ver las capturas de pantalla:

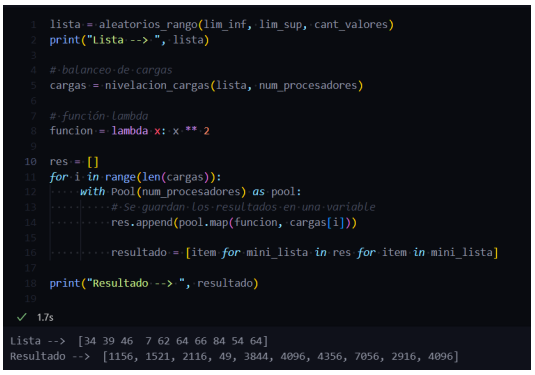

Figure 1: Evaluación paralela con la función lambda x^2

5 Conclusiones

El problema propuesto pudo ser resulelto satisfactoriamente, hubo algunos problemas a la hora de pasar la función lambda como parámetro cuando se intentaba realiar la paralelización; sin embargo, fue mínimo, en general la práctica fue sencilla ya que todo lo hemos visto en clase y en otras materias que se cursaron previamente. Como vimos en algunas practicas pasadas, el tiempo de ejecución se reduce considerablemente a comparación de hacer todo de manera secuencial (aunque esto se note más en conjuntos de datos más grandes), los resultados colocados en este documento fueron pocos debido al espacio disponible; sin embargo, en el programa que se adjuntará tendrá valores mucho más elevados para que se pueda observar bien.