|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | **PRÁCTICA DE LABORATORIO** | |
| **CARRERA**: INGENIERIA DE SISTEMAS/COMPUTACION | | | | **ASIGNATURA**: COMPUTO PARALELO |
| **NRO. PRÁCTICA**: |  | **TÍTULO PRÁCTICA**: Examen de recuperación practico | | |
| **OBJETIVO**  Implementación de multiplicación de matriz por vector utilizando el módulo multiprocessing. | | | | |
| **INSTRUCCIONES** | | | 1. Crear un script en Python utilizando spyder o jupyter e identificarlo con sus nombres completos, en el siguiente formato: *Apellidos\_Nombre\_Examen*. Por ejemplo: LeonParedes\_Gabriel\_Examen.py. 2. Desarrollar el examen. 3. Subir al AVAC: 4. Informe detallado de la resolución del examen en formato .pdf 5. Script de pyhton en formato (.py o .ipynb) | |
|  | | |  | |

|  |
| --- |
| **DESARROLLO** |
| **Enunciado.** El siguiente código secuencial implementa el producto de una matriz B de dimensión N × N por un vector c de dimensión N.    Ilustración 1 Enunciado del algoritmo  **Implementación del algoritmo secuencial:**  import numpy as np  import time  from random import randint  from numpy import savetxt  N = 160  A = np.random.randint(60, size=N)  B = np.random.randint(60, size=(N,N))  C = np.random.randint(60, size=N)  start = time.time()  for i in range(N):  sum = 0  for j in range(N):  sum += B[i][j] \* C[j]  A[i] = sum  print("Tiempo Secuencial: ", (time.time() - start))  savetxt('resultadoSecuencial.csv', A, delimiter=' ')  **implementación algoritmo con procesos:**  Para poder realizar procesos con multiprocesos es indispensable importar el módulo multiprocessing y de este también Process para poder ejecutar la multiplicación usando diferentes procesos.  Primeramente, importamos el módulo numpy para poder generar números aleatorios y llenar la matriz y los vectores.   * N: almacena la dimensión de la matriz y los vectores * A: vector en donde se guardará el resultado de la multiplicación de la matriz B por el vector C * B: matriz de NxN. * C: Vector de tamaño N.   N = int(16000)  A = np.random.randint(60, size=N)  B = np.random.randint(60, size=(N,N))  C = np.random.randint(60, size=N)  El algoritmo cuenta con 2 funciones, la primera función recorre la matriz hasta la mitad y realiza el proceso de multiplicar la mitad de la matriz B por la mitad del vector C:  def multiplicar(B, C, N1, N2):  for i in range(int(N1), int(N2)):  sum2 =0  for j in range(int(N1), int(N2)):  sum2 +=B[i][j]\*C[j]  A[i] = sum2  A esta función le pasamos 4 parámetros:   1. Matriz B 2. Matriz C 3. N1 4. N2   Por último, generamos 5 procesos por separado a los cuales les asignamos la función multiplicar() para multiplicar la matriz B por el vector C, y le definimos un punto de inicio y un punto final.  Para esto definimos un proceso p1 hasta p5 de tipo Process, y le asignamos como target a la función multiplicar y como args le pasamos la matriz B, vector C y el valor de N1 y N2 que serán el inicio y el final de los bucles, como en este caso queremos dividir la matriz en 5 partes iguales, entonces al primer proceso se le asigna y , al segundo proceso y y así consecutivamente hasta llegar a .  Luego mandamos a ejecutar los procesos con los métodos start() y join() para terminar los procesos una vez han realizado las operaciones de multiplicar la matriz B por el vector C.  if \_\_name\_\_ == '\_\_main\_\_':  tInit = time.time()  p1 =Process(target=multiplicar, args=(B, C, int(0), int(N\*(1/5),) ))  p2 =Process(target=multiplicar, args=(B, C, int(N\*(1/5)), int(N\*(2/5),) ))  p3 =Process(target=multiplicar, args=(B, C, int(N\*(2/5)), int(N\*(3/5),) ))  p4 =Process(target=multiplicar, args=(B, C, int(N\*(3/5)), int(N\*(4/5)), ))  p5 =Process(target=multiplicar, args=(B, C, int(N\*(4/5)), int(N), ))  p1.start()  p2.start()  p3.start()  p4.start()  p5.start()  p1.join()  p2.join()  p3.join()  p4.join()  p5.join()  print("Tiempo procesos: ", (time.time()-tInit))  print("N: ", N)  print('processor count: ',multiprocessing.cpu\_count())  savetxt('resultadoOperaciones.csv', A, delimiter='[]')  **Comparativas de tiempo de ejecución:**  Las comparativas de tiempo se las hizo en milisegundos haciendo uso del módulo time.  Ilustración 2 Comparativas tiempos de ejecución  **Cálculos**  **Aceleración:**  **Con N = 10,000:**  **Con N = 20,000:**  **Con N = 25,000:**  **Eficiencia:**  **Con N = 10,000:**  **Con N = 20,000:**  **Con N = 25,000:**  **Conclusiones:**  En este caso se ha obtenido una eficiencia E < 1 por lo que se puede clasificar a este problema como un caso real mas no linear, el caso que más se acercó a 1 es cuando se probó con un . |
|  |

|  |
| --- |
| **RESULTADO(S) OBTENIDO(S)**:   * Implementar un algoritmo por procesos y comparar sus ventajas sobre algoritmos secuenciales. |
| **CONCLUSIONES**:   * El algoritmo por procesos es significativamente más eficiente y veloz al momento de procesar grandes volúmenes de información al compararlo con un algoritmo secuencial |
| **RECOMENDACIONES**:   * Revisar el contenido teórico de la materia. * Medir tiempos de ejecución. |

***Docente*:** Ing. Gabriel León, PhD. Estudiante: Alex Reinoso

***Firma*: Firma: **