**OBSERVACIONES LABORATORIO 7**

Jacobo Zarruk, 202223913

María José Amorocho,202220179

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | AMD Ryzen 5 5625U with Radeon Graphics 2.30 GHz | Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.10 GHz |
| Memoria RAM (GB) | 16.0 GB | 8,00 GB |
| Sistema Operativo | Windows 11 Home Single Language | Windows 10 Pro – 64 bits |

Tabla 1. Especificaciones de las máquinas para ejecutar las pruebas de rendimiento.

# **Maquina 1**

## **Resultados**

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo PROBING*** | | |
| **Factor de Carga (PROBING)** | **Consumo de Datos [kB]** | **Tiempo de Ejecución Real @LP [ms]** |
| 0.1 |  |  |
| 0.5 |  |  |
| 0.7 |  |  |
| 0.9 |  |  |

Tabla 2. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando PROBING en la Maquina 1.

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo CHAINING*** | | |
| **Factor de Carga (CHAINING)** | **Consumo de Datos [kB]** | **Tiempo de Ejecución Real @SC [ms]** |
| 2.00 |  |  |
| 4.00 |  |  |
| 6.00 |  |  |
| 8.00 |  |  |

Tabla 3. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando CHAINING en la Maquina 1.

## **Graficas**

La gráfica generada por los resultados de las pruebas de rendimiento en la **Maquina 1.**

* Comparación de memoria y tiempo de ejecución para PROBING y CHAINING

# **Maquina 2**

## **Resultados**

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo PROBING*** | | |
| **Factor de Carga (PROBING)** | **Consumo de Datos [kB]** | **Tiempo de Ejecución Real @LP [ms]** |
| 0.1 | 3531,66 | 179709,09 |
| 0.5 | 3531,65 | 58644,27 |
| 0.7 | 3531,65 | 93826,36 |
| 0.9 | 3531,65 | 84233,80 |

Tabla 4. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando PROBING en la Maquina 2.

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo CHAINING*** | | |
| **Factor de Carga (CHAINING)** | **Consumo de Datos [kB]** | **Tiempo de Ejecución Real @SC [ms]** |
| 2.00 | 3502,57 | 65300,68 |
| 4.00 | 3504,47 | 54212,80 |
| |6.00 | 3506,10 | 54328,05 |
| 8.00 | 3506,10 | 54949,62 |

Tabla 5. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando CHAINING en la Maquina 2.

## **Graficas**

La gráfica generada por los resultados de las pruebas de rendimiento en la **Maquina 2.**

* Comparación de memoria y tiempo de ejecución para PROBING y CHAINING

# **Preguntas de análisis**

1. ¿Por qué en la función **getTime()** se utiliza **time.perf\_counter()** en vez de otras funciones como **time.process\_time()**?

Mientras que time.perf\_counter() devuelve el valor absoluto de un contador de alto rendimiento, time.process\_time()es una función que retorna la suma del tiempo de CPU del kernel y del espacio de usuario del proceso. Se usa time.perf\_counter()porque es una función que permite saber el tiempo de ejecución total de un programa, pero no se enfoca calcular el tiempo del sistema y de la CPU durante el proceso (pues son datos que dependen mucho más de la máquina que se esté usando).

1. ¿Por qué son importantes las funciones **start()** y **stop()** de la librería **tracemalloc**?

Start() y stop() son funciones que permiten establecer un punto de partida y fin para poder tomar la memoria consumida en un proceso específico. Con start()comienza a rastrear las asignaciones de memoria de Python, y con stop() se detiene este proceso, por lo que se limpia el registro de rastreos después de tomar una captura de los datos. Sin estas funciones no se puede iniciar ni detener la toma de datos de memoria consumida, pues son el punto de referencia que el resto de API’s en la librería usan para su funcionamiento.

1. ¿Por qué no se puede medir paralelamente el **uso de memoria** y el **tiempo de ejecución** de las operaciones?

No es posible medir simultáneamente el uso de memoria y el tiempo de ejecución, ya que justamente medir el uso de memoria requiere tiempo adicional de ejecución. Si se midiesen paralelamente, el tiempo obtenido sería la suma del tiempo que tarda el código en sí mas el tiempo que se demora el código en obtener la memoria utilizada.

1. Teniendo en cuenta cada uno de los requerimientos del reto ¿Cuántos índices implementaría en el Reto? y ¿Por qué?

Teniendo presentes los requerimientos del reto, consideramos que sería bueno implementar un índice de “año”, así como se hizo en la práctica. No creemos que sean necesarios más índices en la estructura de datos principal pues en todos los requerimientos se pide organizar la información principalmente por año; sin embargo, sí es necesario implementar nuevas estructuras para ordenar la información de acuerdo con los sectores y subsectores económicos.

1. Según los índices propuestos ¿en qué caso usaría **Linear Probing** o **Separate Chaining** en estos índices? y ¿Por qué?

Se implementaría Separate Chaining para optimizar un poco el consumo de memoria.

1. Dado el número de elementos de los archivos del reto (large), ¿Cuál sería el factor de carga para estos índices según su mecanismo de colisión?

Teniendo en cuenta que el archivo large contiene 4903 actividades económicas, si se usa linear probing, el factor de carga debería ser menor o igual a 0.5; es decir, que como máximo la tabla tenga el doble de espacios que el numero de elementos a insertar en ella. Si se usa Separate Chaining lo óptimos sería manejar un factor de carga cercano a 1.

1. ¿Qué cambios percibe en el **tiempo de ejecución** al modificar el factor de carga máximo para cargar el catálogo de contenido Streaming?

Al aumentar el factor de carga, el tiempo de ejecución fue menor usando ambos tipos de manejos de colisiones.

1. ¿Qué cambios percibe en el **consumo de memoria** al modificar el factor de carga máximo para cargar el catálogo de contenido Streaming?

Tanto para el manejo de colisiones usando Linear Probing y Separate Chaining, mientras que el factor de carga aumentaba, el consumo de memoria permanecía prácticamente constante.

1. ¿Qué cambios percibe en el **tiempo de ejecución** al modificar el esquema de colisiones?, si los percibe, describa las diferencias y argumente su respuesta.

Al modificar el esquema de colisiones pudimos observar que con el uso de Separate Chaining el tiempo de ejecución fue considerablemente menor que usando Linear Probing. En promedio, usando Separate Chaining el programa demoró 46’905ms menos que con el otro método.

1. ¿Qué cambios percibe en el **consumo de memoria** al modificar el esquema de colisiones?, si los percibe, describa las diferencias y argumente su respuesta.

Al usar Separate Chaining el consumo de memoria es menor que cuando se ejecuta el programa con Linear Probing. Aunque no es muy notoria la diferencia, mientras que en promedio Separate Chaining consume 3504 kB, Linear Probing usa alrededor de 3531 kB.

1. ¿Qué configuración de ideal ADT Map escogería para el **índice de años (“Año”)** ?, especifique el mecanismo de colisión, el factor de carga y el numero inicial de elementos.